UPV

Sistema bajo coste basado en FPGA para la adquisición de datos de sensores con monitorización inalámbrica sobre dispositivos móviles.

Master Universitario en Ingeniería de Sistemas Electrónicos.

AUTOR: IGNACIO M. BLASCO LAHOZ.

TUTOR: RICARDO JOSE COLOM PALERO.

11/09/2017

1. Contenido

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

2. Introducción.

En el mundo de hoy en día se está extendiendo el uso de comunicaciones inalámbricas para todo tipo de usos. Esto usos, se pueden ver diariamente cuando vas a correr y tu pulsera cuenta pasos comunica a tu móvil la evolución de tu deporte diario o de una manera más profesional en la monitorización del regadío de una plantación agraria. Además, de este auge de las comunicaciones inalámbricas, también evolucionando positivamente el desarrollo de sistema integrados donde con un mismo dispositivo se llega a desarrollar o diseñar casi todo el sistema, tales como los sistemas basados en Fpga.

Así pues si consideramos que somos estudiantes, que lo somos, o que no tenemos un gran poder adquisitivo, que no lo tenemos, el poder usar una herramienta asequible económicamente que nos permita diseñar de arriba a abajo todo nuestro sistema, tanto software como hardware y que además tengamos la capacidad de utilizar tecnologías punteras, permite la capacidad de desarrollar productos finales que de otra manera no estarían a nuestro alcance.

2.1. Objetivo.

- Comunicar sistemas inteligentes, mediante tecnologías inalámbricas, a sistemas de monitorización.
- La realización de un TFM, con implementación hardware utilizando recursos al alcance de cualquiera por un precio bajo. Sin recurrir al uso de equipos de laboratorio costosos, a los cuales no todo el mundo tiene acceso. Esto entraría dentro de la filosofía americana, de las grandes empresas que surgieron en garajes con pocos recursos.

2.2. Actualidad en la materia.

Como ya se ha comentado en el preámbulo de la introducción, existen una gran cantidad de aplicaciones donde habitan las tecnologías inalámbricas de una manera exitosa. A continuación, vamos a poner unos ejemplos de estos usos y de los productos que existen en el mercado para satisfacerlos.

2.2.1. Domótica / Home Automation.

Es una de las aplicaciones más usadas con tecnologías inalámbricas, ya que es muy fácil la instalación de dispositivos y la modificación de la posición de los mismos. Los usos típicos son:

Seguridad:

Sensores de movimiento, de rotura de cristales, apertura de puertas y ventanas.

A pesar de su baja velocidad también se usa para transmitir imágenes de cámara de seguridad de baja calidad.

Lectura de instrumentos de servicios:

Los medidores de consumo de agua, gas y energía eléctrica deben leerse de forma regular a efecto de facturar los servicios. También los medidores inalámbricos

podrían comunicarse con los artefactos dentro de la casa. Por ejemplo ante un pico de consumo eléctrico se podría desconectar algún equipo de alto consumo.

Sistema de riego automático:

El uso de un medidor de humedad de suelo permite mejorar la eficiencia del consumo de agua. Se puede distribuir una red de sensores de humedad en un parque de modo que solo se riegue las zonas secas y controlar el tiempo de regado.

Control de iluminación:

Para poder controlar el encendido de una lámpara se necesita un cableado a una llave interruptora en una caja de una pared. Con la tecnología inalámbrica se simplifica la instalación de controles evitando tener que pasar un cable. Aunque el coste de la conexión inalámbrica es más elevado que el convencional cableado. Es posible conectar un controlador inteligente que encienda/apague luces de acuerdo a una programación, la detección de presencia de personas o algún otro criterio.

Control de temperatura multizona:

Los termostatos se usan para controlar la temperatura de una casa. En los sistemas de aire acondicionado es posible controlar las rejillas deflectoras amortiguadora de aire de modo de tener control de temperatura separado para cada habitación.

Controles remotos:

Tradicionalmente los controles remotos de TV, DVD y equipos de audio usan tecnología óptica infrarroja cuya limitación más importante es que solo funciona a muy poca distancia y sin obstáculos. No puede, por ejemplo, penetrar una pared. Además la comunicación es unidireccional. Usando radiofrecuencia, desaparecen estas limitaciones. Por ejemplo, alguien puede desde otra habitación manejar el equipo de audio y recibir en un display del control remoto, datos de la canción que está escuchando.

2.2.2. Salud.

El campo de la salud ofrece un gran número de oportunidades para las aplicaciones inalámbricas entre las cuales podemos destacar 3 áreas principales:

- Seguimiento de enfermedades crónicas (Chronic Disease Monitoring).
- Supervisión del bienestar personal (Personal Wellness).
- Personal fitnes.

Seguimiento de enfermedades crónicas (Aplications for health care):

Hay personas con enfermedades, que requieren de un control continuo como la diabetes, el asma, enfermedades del corazón o trastornos del sueño.

Existe toda una red de comunicaciones alrededor del paciente, desde dispositivos instalados en el propio cuerpo de la persona, hasta el teléfono móvil del médico o un familiar del paciente, para tener ese control sobre la persona.

Seguimiento del bienestar personal (Personal Wellness Monitoring):

Los dispositivos destinados a este tema están diseñados para ser utilizados por personas mayores, para que puedan garantizar su salud, su seguridad, su bienestar y su localización. La información es transmitida gracias red inalámbrica a un centro de control donde es procesada y analizada.

Personal fitness:

Habitual en gimnasios i en hogares, vigilan el ritmo cardiaco, las calorías quemadas durante el ejercicio, la temperatura corporal o el nivel de oxígeno en sangre. La posibilidad de usar m_ultiples sensores, gracias al bajo consumo y precio de éstos, permite controlar diferentes tipos de variables durante periodos de tiempo distintos. Es decir, variables como el ritmo card__aco se controlar_an cada menos tiempo que otras caracter__sticas como la temperatura.

Además existen muchos otros mercados como los siguientes:

- Machine 2 / Machine (M2M).
- Smart Metering.
- Agricultura de Precisión.
- Ciudades Inteligentes.

2.2.3. Mercado.

Obviamente, no todas las tecnologías inalámbricas sirven para lo mismo, ya que cada una tiene su propio punto fuerte. Por poner un ejemplo, es más usual utilizar Bluetooth para comunicar punto a punto una cantidad media de información, tal como el envío de música a unos auriculares inalámbricos.

Esta distinción, se verá más claro en el apartado de tecnologías inalámbricas posterior, eso sí, vamos a ver la tecnología inalámbrica como un conjunto y vamos a hacer diferentes distinciones teniendo en cuenta otras características.

Específicos:

Existen productos diseñados a “Dock” para cubrir aplicaciones específicas, tales como las ya mencionadas. Ejemplos de productos de estas características podríamos decir las referentes a las siguientes empresas:

Ilustración 1: Logotipo de la empresa Delta OHM

Ilustración 2: Logotipo de la empresa Masimo.

Estas empresas son lo que cualquier diseñador estaría deseando alcanzar, la posibilidad de producir productos específicos que solucionan problemas específicos. Estos productos, dentro de lo específicos que son, tienen un abanico de precio bastante aceptable.

Además, existen productos con una sola función pero que se han generalizado de tal manera que se ha podido bajar los precios de manera sustancial.

- Productores de wereables, tecnologías para conexión a internet entre otros.

Genéricos:

A su vez, podríamos ampliar los tipos de productos a genéricos en el caso de entender como tales las plataformas económicas para desarrollar un producto específico. Estos productos podríamos diferenciarlos de tres maneras:

- ***Metodología mediante "Micro-pc",*** donde el sistema se realiza en un sistema operativo al uso y se necesita una placa de ampliación para la comunicación. Ejemplo de este uso, podría ser a través de la tarjeta de BeagleBone Black y algún tipo de tarjeta de ampliación que contenga un módulo de comunicación. El problema de esta metodología es la incapacidad de modificar hardware de manera rápida.
- ***Metodología mediante Microcontrolador***, el sistema sería similar al anterior solamente que a priori no debería correr en un sistema operativo y mantendría el mismo problema o restricción.
- ***Metodología mediante Fpga***, Esta es la manera que se va usar en este documento, esta metodología lo permite en cierta manera todo. Esto se debe a que podemos implementar un microprocesador en la fpga y en ésta un sistema operativo o incluso puede ya estar incorporado de manera física, además obviamente tenemos el grado de libertad de poder modificar el hardware, de la fpga, si fuera necesario.

Laboratorio:

Fuera de la filosofía de producir un producto lo más económicamente eficiente, incluso a expensas de la calidad de la medición, existe los productos de laboratorio

que sí que cumplen la premisa de obtener mediciones exhaustivas en tiempo y forma.

2.3. Hardware seleccionado.

Como ya se ha comentado anteriormente la metodología seguida consiste en utilizar una tarjeta de desarrollo con fpga y una tarjeta de ampliación para darle la capacidad de comunicación inalámbrica. Así pues, vamos a describir este hardware.

2.3.1. DE0_Nano.

Es una tarjeta de Terasic para desarrollo y aprendizaje de proyectos no pesados con base en FPGA. Las características reseñables a mencionar son las siguientes:

- Su bajo coste de adquisición, 61 o 79$ dependiendo de la modalidad de compra.
- Su pequeño tamaño.
- Posibilidad de alimentación propia.

Estas características posibilitan su uso para diseños portables y de bajo coste, como robóticas, de monitorización etc…

Además, es interesante comentar las características de la tarjeta con más detenimiento:

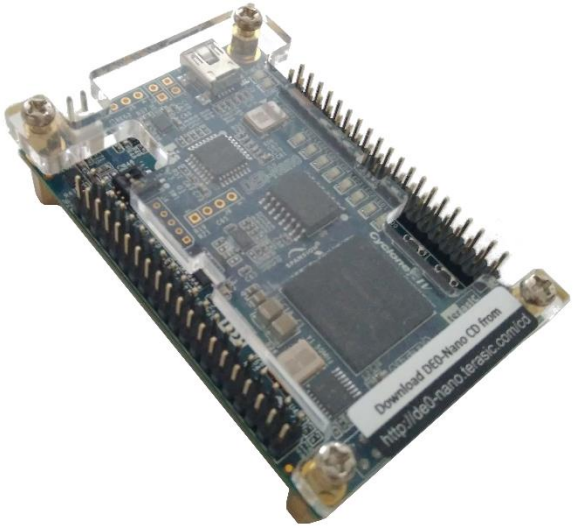


Ilustración 3: Tarjeta DE0-Nano

Características principales:

Featured device:

- Altera Cyclone® IV EP4CE22F17C6N FPGA
- 153 maximum FPGA I/O pins

Configuration status and set-up elements:

- On-board USB-Blaster circuit for programming
- Spansion EPCS64

Expansion header:

- Two 40-pin Headers (GPIOs) provide 72 I/O pins, 5V power pins, two 3.3V power pins and four ground pins

Memory devices:

- 32MB SDRAM
- 2Kb I2C EEPROM

General user input/output:

- 8 green LEDs
- 2 debounced pushbuttons

- 4-position DIP switch

G-Sensor:

- ADI ADXL345, 3-axis accelerometer with high resolution (13-bit)

A/D Converter:

- NS ADC128S022, 8-Channel, 12-bit A/D Converter
- 50 Ksps to 200 Ksps

Clock system:

- On-board 50MHz clock oscillator

Power Supply:

- USB Type mini-AB port (5V)
- DC 5V pin for each GPIO header (2 DC 5V pins)
- 2-pin external power header (3.6-5.7V)

2.3.2. RFS.

Es una tarjeta de ampliación también de Terasic, que será la que entre otras cosas nos brindará la comunicación inalámbrica mediante “Wi-Fi” y “Bluetooth” a algún soporte de monitorización.

Características principales:

- 2x20 GPIO interface.
- CWi-Fi, using ESP- WROOM -02 module.
- Bluetooth SPP, using HC-05 module.
- Ambient Light Sensor.
- Temperature and humidity sensor.
- 9-axissensor –accelerometer, magnetometer, and gyroscope.
- UARTto USB.
- 2x6 TMD GPIO Header.

Ilustración 4: Tarjeta RFS

Características específicas de los módulos de comunicación:

Módulo ESP-WROOM-02 (Wi-Fi):

Tabla 1: Características ESP-WROOM-02

Categories	Items	Specifications
Wi-Fi	Standards	FCC/CE/TELEC/KCC/LCIE/IC/NCC
	Wi-Fi protocols	802.11 b/g/n
	Frequency range	2.4 GHz – 2.5 GHz (2400 M – 2483.5 M)
Hardware	Peripheral interface	UART/HSPI/I2C/I2S/IR Remote Control
		GPIO/PWM
	Operating voltage	3.0 V – 3.6 V
	Operating current	Average: 80 mA
	Operating temperature range	-40°C – 85°C
	Storage temperature	-40°C – 85°C
	Package size	18 mm x 20 mm x 3 mm
	External interface	-
	Wi-Fi mode	Station/SoftAP/SoftAP + Station

Categories	Items	Specifications
Software	Security	WPA/WPA2
	Encryption	WEP/TKIP/AES
	Firmware upgrade	UART Download / OTA (via network) / Download and write firmware via host
	Software development	Supports Cloud Server Development / SDK for custom firmware development
	Network protocols	IPv4, TCP/UDP/HTTP/FTP
	User configuration	AT Instruction Set, Cloud Server, Android/iOS App

Módulo HC-05 (Bluetooth):

- *Wireless transceiver:*
  - Sensitivity (Bit error rate) can reach -80dBm.
  - The change range of output's power: -4 - +6dBm.
- *Function description (perfect Bluetooth solution):*
  - Has an EDR module; and the change range of modulation depth: 2Mbps - 3Mbps.
  - Has a build-in 2.4GHz antenna; user needn't test antenna.
  - Has the external 8Mbit FLASH
  - Can work at the low voltage (3.1V~4.2V). The current in pairing is in the range of 30～40mA. The current in communication is 8mA.
  - PIO control can be switched.
  - Has the standard HCI Port (UART or USB)
  - The USB protocol is Full Speed USB1.1, and compliant with 2.0.
  - This module can be used in the SMD.
  - It's made through RoHS process.
  - The board PIN is half hole size.
  - Has a 2.4GHz digital wireless transceiver.
  - Bases at CSR BC04 Bluetooth technology.
  - Has the function of adaptive frequency hopping.
  - Small (27mm×13mm×2mm).
  - Peripheral circuit is simple.
  - It's at the Bluetooth class 2 power level.
  - Storage temperature range: -40 ℃ - 85℃, operating temperature range: -25 ℃ - +75℃

- Any wave inter Interference: 2.4MHz，the power of emitting: 3 dBm.
- Bit error rate: 0. Only the signal decays at the transmission link, bit error may be produced. For
- example, when RS232 or TTL is being processed, some signals may decay.

- *Low power consumption.*
- *Has high-performance wireless transceiver system.*
- *Low Cost.*
- *Application fields:*
  - Bluetooth Car Handsfree Device Bluetooth GPS
  - Bluetooth PCMCIA , USB Dongle
  - Bluetooth Data Transfer
- *Software:*
  - CSR

Características específicas de los módulos de sensores:

Ambient Light Sensor (APDS-9301):

- Approximate the human-eye response
- Precise Illuminance measurement under diverse lighting conditions
- Programmable Interrupt Function with User-Defined Upper and Lower Threshold Settings
- 16-Bit Digital Output with I^2C Fast-Mode at 400 kHz
- Programmable Analog Gain and Integration Time
- Miniature ChipLED Package
  - Height – 0.55mm
  - Length – 2.60mm
  - Width – 2.20mm
- 50/60-Hz Lighting Ripple Rejection
- Typical 3.0V Input Voltage
- Low Active Power (0.6 mW Typical) with Power Down Mode
- RoHS Compliant

Temperature and humidity sensor HDC1000:

- Relative Humidity (RH) Operating Range 0% to 100%.
- 14 Bit Measurement Resolution.

- Relative Humidity Accuracy ±3%.
- Temperature Range.
    - Operating -20°C to 85°C
    - Functional -40°C to 125°C
- Temperature Accuracy ±0.2°C
- 200nA Sleep Mode Current
- Average Supply Current:
    - 820nA @ 1sps, 11 bit RH Measurement.
    - 1.2µA @ 1sps, 11 bit RH and Temperature Measurement
- Tiny 2mm x 1.6mm Device Footprint
- I2C Interface
- Supply Voltage 3V to 5V

9-axissensor –accelerometer, magnetometer, and gyroscope (MPU-9250)

➢ *Gyroscope Features:*

The triple-axis MEMS gyroscope in the MPU-9250 includes a wide range of features:

- Digital-output X-, Y-, and Z-Axis angular rate sensors (gyroscopes) with a user-programmable full-scale range of ±250, ±500, ±1000, and ±2000°/sec and integrated 16-bit ADCs
- Digitally-programmable low-pass filter
- Gyroscope operating current: 3.2mA
- Sleep mode current: 8µA
- Factory calibrated sensitivity scale factor
- Self-test

➢ *Accelerometer Features:*

The triple-axis MEMS accelerometer in MPU-9250 includes a wide range of features:

- Digital-output triple-axis accelerometer with a programmable full scale range of ±2*g*, ±4*g*, ±8*g* and ±16*g* and integrated 16-bit ADCs
- Accelerometer normal operating current: 450µA
- Low power accelerometer mode current: 8.4µA at 0.98Hz, 19.8µA at 31.25Hz
- Sleep mode current: 8µA
- User-programmable interrupts
- Wake-on-motion interrupt for low power operation of applications processor

- Self-test

➢ *Magnetometer Features:*

The triple-axis MEMS magnetometer in MPU-9250 includes a wide range of features:

- 3-axis silicon monolithic Hall-effect magnetic sensor with magnetic concentrator
- Wide dynamic measurement range and high resolution with lower current consumption.
- Output data resolution of 14 bit (0.6μT/LSB) or 16 bit (15μT/LSB)
- Full scale measurement range is ±4800μT
- Magnetometer normal operating current: 280μA at 8Hz repetition rate
- Self-test function with internal magnetic source to confirm magnetic sensor operation on end products

➢ *Additional Features*

The MPU-9250 includes the following additional features:

- Auxiliary master I^2C bus for reading data from external sensors (e.g. pressure sensor)
- 3.5mA operating current when all 9 motion sensing axes and the DMP are enabled
- VDD supply voltage range of 2.4 – 3.6V
- VDDIO reference voltage for auxiliary I^2C devices
- Smallest and thinnest QFN package for portable devices: 3x3x1mm
- Minimal cross-axis sensitivity between the accelerometer, gyroscope and magnetometer axes
- 512 byte FIFO buffer enables the applications processor to read the data in bursts
- Digital-output temperature sensor
- User-programmable digital filters for gyroscope, accelerometer, and temp sensor
- 10,000 *g* shock tolerant
- 400kHz Fast Mode I^2C for communicating with all registers
- 1MHz SPI serial interface for communicating with all registers
- 20MHz SPI serial interface for reading sensor and interrupt registers
- MEMS structure hermetically sealed and bonded at wafer level
- RoHS and Green compliant

- *MotionProcessing*
  - Internal Digital Motion Processing™ (DMP™) engine supports advanced MotionProcessing and low power functions such as gesture recognition using programmable interrupts
  - Low-power pedometer functionality allows the host processor to sleep while the DMP maintains the step count.

2.3.3. Móvil Android.

Es necesario disponer de un móvil con las siguientes características dependiendo de la tecnología de comunicación:

- *Bluetooth:*
  - Mínima versión de android 4.2.2 (Jelly Bean).
  - Máxima versión probada android 5.0.1 (Lollipop)
  - La app de RFS.
  - Conectividad Bluetooth.

Aunque sería relativamente sencillo adaptarlas a otra versión cambiando la sdk en la creación de la app.

- *Wifi:*
  - Conectividad Wifi.
  - Navegador web.

Así pues, se podría monitorizar los sensores y actuar sobre elementos de la tarjeta DE0-nano desde cualquier dispositivo que disponga de estas características, como por ejemplo un pc.

2.4. Software Usado.

El software que se ha utilizado para realizar el diseño, que se implementará en la fpga, es el entorno de desarrollo Quartus Prime. Este entorno de desarrollo, lleva todo un compendio de herramientas para diseñar, comprobar y simular el diseño tanto hardware como software.

- *Quatus Prime*, permite diseñar archivos Verilog, implementar los pines de salidas, etc..
- *Qsys*, permite el desarrollo del hardware relacionado con el micro soft NIOS II.
- *Eclipse*, es el encargado de desarrollar el software del NIOS II.
- *ModelSim*, sirve para tomar mediciones del hardware.
- Etc…

Ilustración 5: Logotipo de Quartus Prime

El otro software necesario para diseñar el sistema es Eclipse, este Entorno de Desarrollo además de formar parte de Quartus Prime para el desarrollo del software de micro soft. de la fpga, ha servido como plataforma de desarrollo para la aplicación Android que servirá como cliente Bluetooth.

Ilustración 6: Logotipo de Eclipse

Es importante remarcar que ha sido necesario instalar el eclipse con una extensión específica para Android y el SDK 5.01:

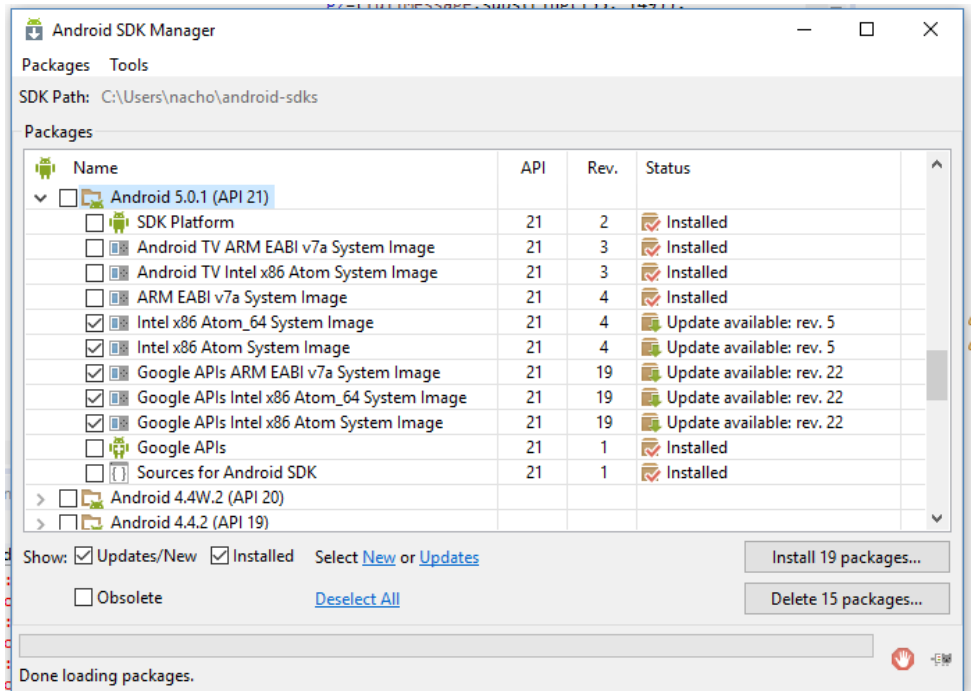


Ilustración 7: Android SDK manager.

2.5. Metodología de trabajo del TFM.

Podemos estructurar la manera en que se ha trabajado, mediante dos partes bien diferenciadas.

La primera parte corresponde a la adaptación de los archivos ejemplo, proporcionados por Terasic, para otro tipo de placas:

- Se suministran en el archivo adjuntado como “RFS_v.1.2.0_SystemCD”.

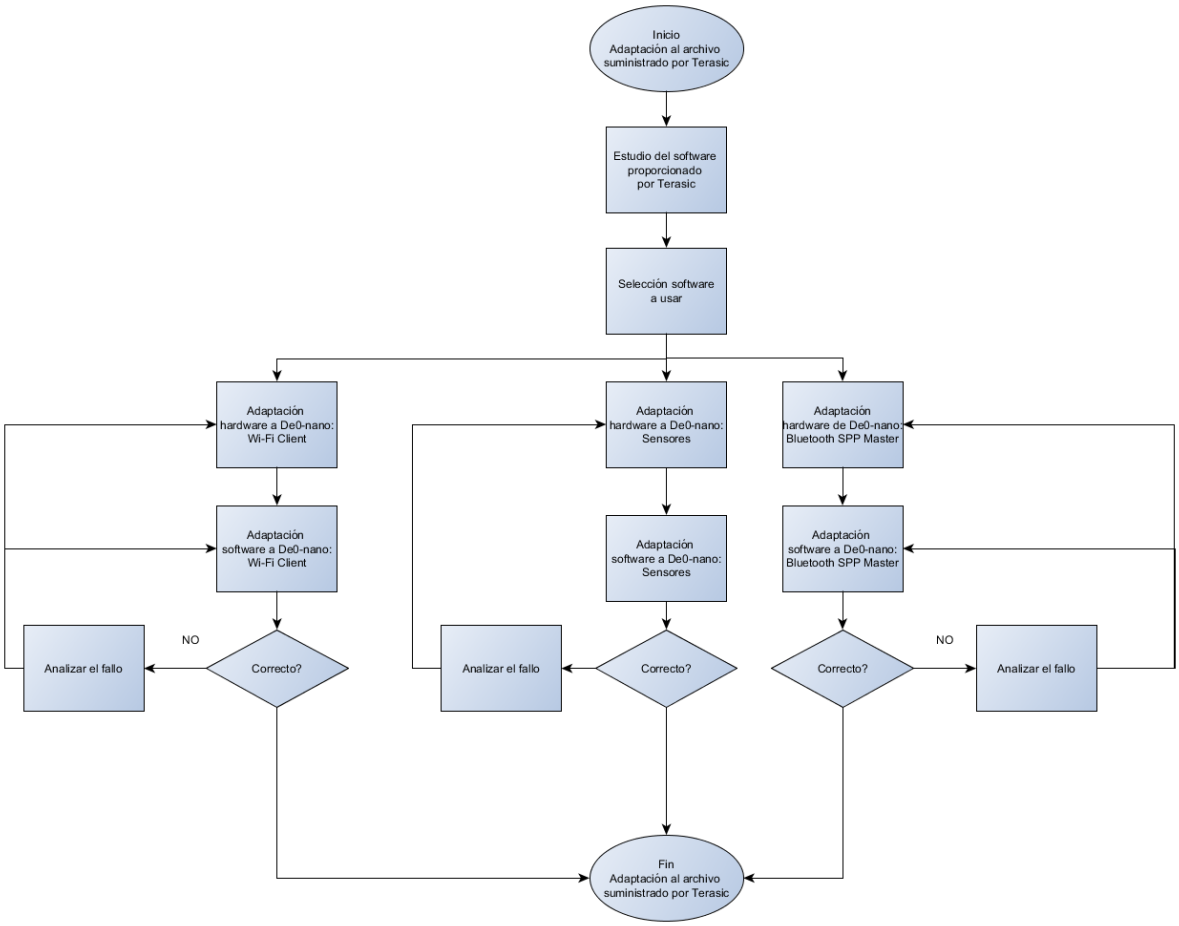


Ilustración 8: Diagrama flujo metodología de trabajo, primera parte.

Como se puede observar el procedimiento consiste en reproducir el sistema para la placa DE0-nano y una vez realizado comprobar si funciona, en caso contrario analizar el fallo ya sea mediante alguna herramienta de Debug incorporada en el software o mediante análisis de las posibles causas. Haciendo esto se ha encontrado algún tipo de error de versiones de software, entre otros inconvenientes, que se comentarán en otros apartados. También comentar que no ha sido necesario el uso de herramientas de comprobación de las señales en el hardware, como “ModelSim”, ya que la adaptación ha sido bastante fluida.

La segunda parte consiste en mejorar el archivo ya adaptado:

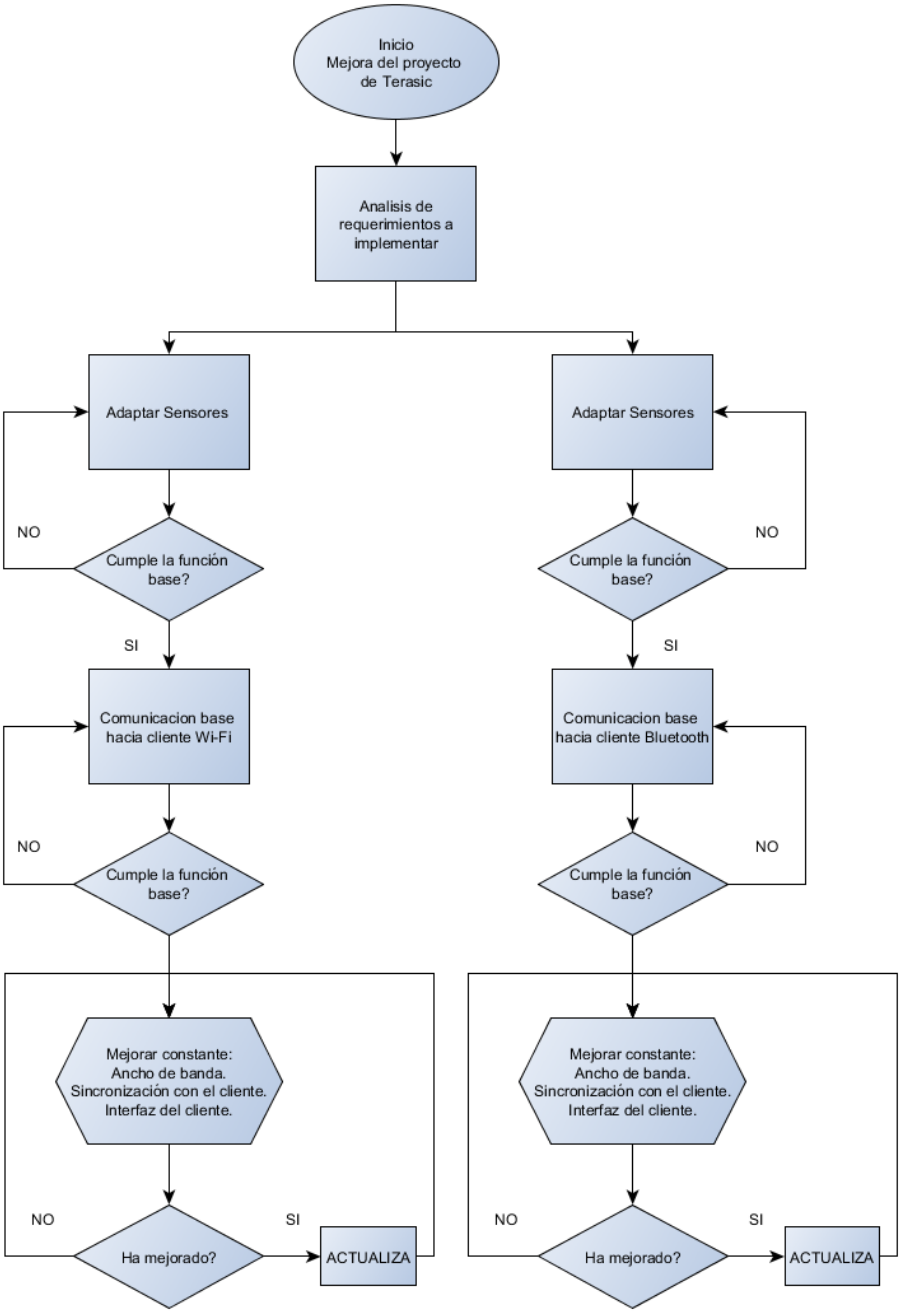


Ilustración 9: Diagrama flujo metodología de trabajo, segunda parte.

Esta segunda parte, ha consistido por una parte en incluir la funcionalidad de la lectura de los sensores a cada una de las dos líneas de desarrollo.

Estas líneas son por una parte la de comunicación por Bluetooth y por otra la de Wi-Fi, una vez se ha incorporado y comprobado, se ha realizado la comunicación de la placa de desarrollo hacia los sistemas de monitorización.

Después de esto el desarrollo se ha basado en mejorar las características del sistema.

3. Conceptos previos.

3.1. Tecnologías inalámbricas.

En el área de la PAN (Personal Area Network) podemos encontrar las siguientes tecnologías:

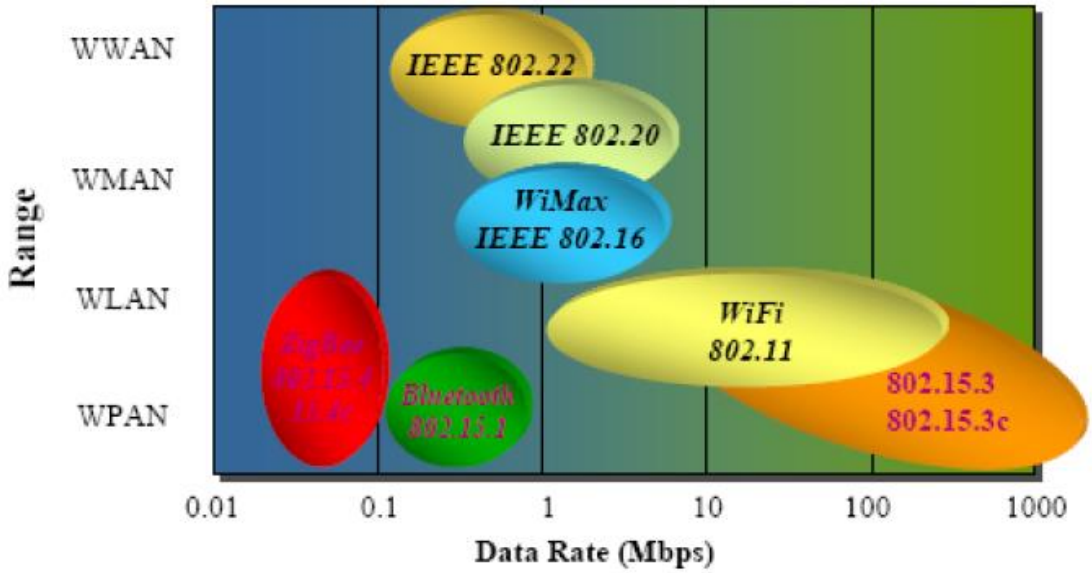


Ilustración 10: Gráfica comparativa entre los distintos protocolos inalámbricos.

3.1.1. Bluetooth.

Bluetooth es una especificación industrial para Redes Inalámbricas de Área Personal (WPANs) que posibilita la transmisión de voz y datos entre diferentes dispositivos mediante un enlace por radiofrecuencia en la banda ISM de los 2,4 GHz. Los principales objetivos que se pretenden conseguir con esta norma son:

- Facilitar las comunicaciones entre equipos móviles y fijos.
- Eliminar cables y conectores entre éstos.
- Ofrecer la posibilidad de crear pequeñas redes inalámbricas y facilitar la sincronización de datos entre equipos personales.
- Usos y aplicaciones
- Se denomina Bluetooth al protocolo de comunicaciones diseñado especialmente para dispositivos de bajo consumo, con una baja cobertura y basados en transceivers de bajo coste.
- Gracias a este protocolo, los dispositivos que lo implementan pueden comunicarse entre ellos cuando se encuentran dentro de su alcance. Las comunicaciones se realizan por radiofrecuencia de forma que los dispositivos no tienen que estar alineados y pueden incluso estar en habitaciones separadas si la potencia de transmisión lo permite. Estos dispositivos se clasifican como "Clase 1", "Clase 2" o "Clase 3" en referencia

a su potencia de transmisión, siendo totalmente compatibles los dispositivos de una clase con los de las otras.

-

Tabla 2: Clasificación de los dispositivos Bluetooth

Clase	**Potencia máxima permitida (mW)**	**Potencia máxima permitida (dBm)**	Rango (aproximado)
Clase 1	100 mW	20 dBm	~100 metros
Clase 2	2,5 mW	4 dBm	~25 metros
Clase 3	1 mW	0 dBm	~1 metro

- En la mayoría de los casos, la cobertura efectiva de un dispositivo de Clase 2 se extiende cuando se conecta a un transceptor de Clase 1. Esto es así gracias a la mayor sensibilidad y potencia de transmisión del dispositivo de Clase 1, es decir, la mayor potencia de transmisión del dispositivo de Clase 1 permite que la señal llegue con energía suficiente hasta el de Clase 2. Por otra parte la mayor sensibilidad del dispositivo de Clase 1 permite recibir la señal del otro pese a ser más débil.
- Los dispositivos con Bluetooth también pueden clasificarse según su ancho de banda:

Tabla 3: Clasificación de los dispositivos Bluetooth según su ancho de banda

Versión	Ancho de banda
Versión 1.2	1 Mbit/s
Versión 2.0 + EDR	3 Mbit/s
UWB Bluetooth (propuesto)	53 - 480 Mbit/s

3.1.2. Ultrawideband.

Ultra-wideband (también UWB) se usa para referirse a cualquier tecnología de radio que usa un ancho de banda mayor de 500 MHz o del 25% de la frecuencia central, de acuerdo con la FCC (*Federal Communications Commission*).

Características UWB:

UWB difiere sustancialmente de las estrechas frecuencias de banda de radio (RF) y tecnologías “spread spectrum” (SS), como el Bluetooth, ZigBee y el 802.11. UWB usa un gran ancho de banda del espectro de RF para transmitir información. Por lo tanto, UWB es capaz de transmitir más información en menos tiempo que las tecnologías anteriormente citadas.

Mientras que Bluetooth, WiFi, teléfonos inalámbricos y demás dispositivos de radiofrecuencia están limitadas a frecuencias sin licencia en los 900 MHz, 2,4 GHz y 5,1 GHz, UWB hace uso de un espectro de frecuencia recientemente legalizado. UWB puede usar frecuencias que van desde 3,1 GHz hasta 10,6 GHz: una banda de más de 7 GHz de anchura. Cada canal de radio tiene una anchura de más de 500 MHz, dependiendo de su frecuencia central.

El hecho de estar compartiendo bandas de frecuencia con otros dispositivos, ha hecho que aunque esto les permite tener una alta productividad, han de estar relativamente cerca.

Las ventajas que ofrece UWB son su bajo consumo (como emisor de ondas de radio), bajo coste (se puede usar tecnología CMOS para implementar un dispositivo UWB radio) y alta productividad, lo que marca esta tecnología como el futuro de las WPAN.

Además, UWB permite reutilización de espectros. Por ejemplo, podemos tener una serie de dispositivos en nuestro salón de casa, comunicándose con nuestro ordenador a través de un canal, y a la vez, en otra habitación, otra serie de dispositivos en el mismo canal comunicándose igualmente. WPAN basadas en UWB pueden hacer uso del mismo canal sin interferencias, debido a los rangos tan cortos que permite UWB.

3.1.3. WIFI.

WiFi, es la sigla de Wireless Fidelity (Wi-Fi), que literalmente significa Fidelidad inalámbrica.

Actualmente existen cuatro tipos de conexiones:

- El primero es el estándar IEEE 802.11b que opera en la banda de 2,4 GHz a una velocidad de hasta 11 Mbps.
- El segundo es el IEEE 802.11g que también opera en la banda de 2,4 GHz, pero a una velocidad mayor, alcanzando hasta los 54 Mbps.
- El tercero, que está en uso es el estándar IEEE 802.11 que se le conoce como WiFi 5, ya que opera en la banda de 5 GHz, a una velocidad de 54 Mbps. Una de las principales ventajas de esta conexión es que cuenta con menos interferencias que los que operan en las bandas de 2,4 GHz ya que no comparte la banda de operaciones con otras tecnologías como Bluetooth o ZigBee.
- El cuarto, es el IEEE 802.11n que opera en la banda de 2,4 GHz a una velocidad de 108 Mbps.

3.1.4. ZigBee.

Esta tecnología consigue unos consumos muy bajos y por lo tanto una esperanza de vida de la batería muy elevada en comparación con las demás tecnologías. Esto se ha conseguido simplificando los nodos y el software.

Resumiendo sus principales ventajas, ya que más adelante explicaremos el protocolo con detalle:

- Redes grandes de hasta 65000 nodos
- Adecuado para aplicaciones de menor ancho de banda que las de Bluetooth.
- Bajo consumo.
- Topologías que soporta: estrella, árbol y malla.

Tabla 4: Comparativa entre las tecnologías

Name	ZigBee	Wi-Fi	Bluetooth	UWB
Standard	802.15.4	802.11b	802.15.1	802.15.3
Application	Monitoring & Control	Wireless LAN, Internet	PC/Mobile phone/PDA connectivity	Streaming Video, Home Entertainment
Bandwidth (Kbps)	20-250 Kbps	11+ Mbps	1Mbps	100-500+ Mbps
Range (meters)	1-100+	1-100	1-10+	10+
Resources	4KB-64KB	1MB+	250KB+	--
Battery Life (days)	100-1000+	0.5-5	1-7 days	--
Success Factors	Battery Life,Cost, Reliability	Speed, Flexibility	Cost, Convenience	Speed, Volume

4. Diseño del sistema.

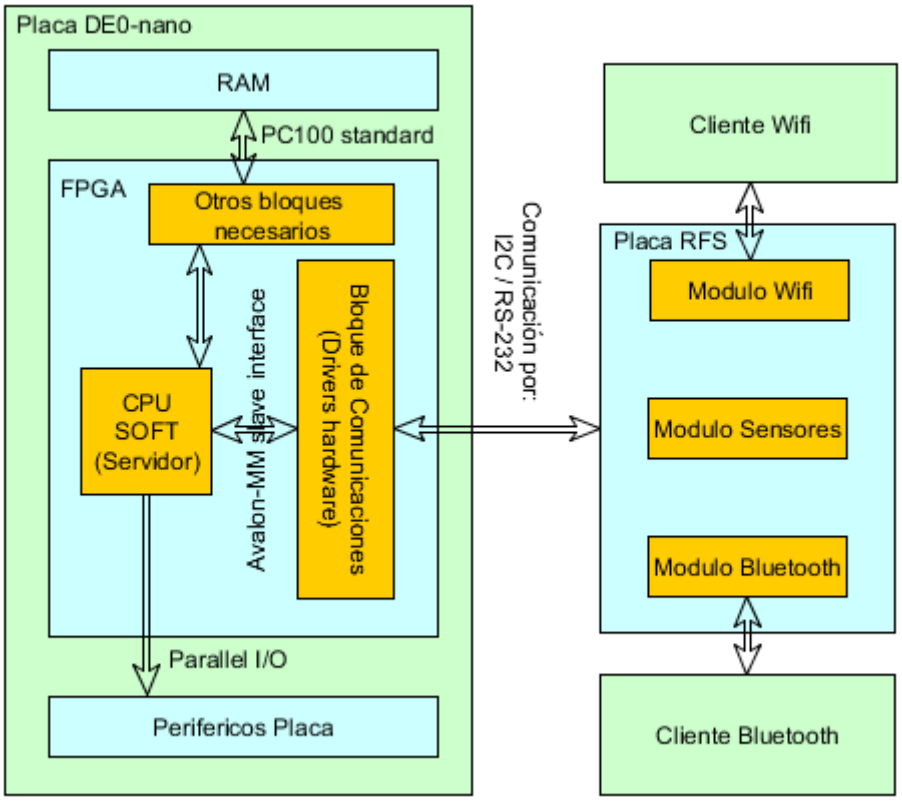


Ilustración 11: Modelo del Sistema.

El sistema que se ha decidido a implementar es el de Cliente/Servidor, es decir el servidor está a la espera de una petición del cliente para responderle en consecuencia.

Para implementar el servidor se va a utilizar la tarjeta de desarrollo ya mencionada con su tarjeta de ampliación. Este tipo de dispositivos brinda la posibilidad de desarrollar el sistema a dock para las necesidades de éste, es decir con una metodología solo hardware o también incorporando un micro soft. lo que permitiría realizar un diseño más flexible. Se ha decidido en implementar un diseño mixto hardaware/software para el servidor, donde el servidor correrá en el micro soft. creado en el hardware de la Fpga y que también permitirá la comunicación con la placa de ampliación a través de comunicación I2C o RS232 dependiendo del módulo a usar.

En cuanto al cliente, tendremos dos dependiendo del tipo de tecnología de transmisión de datos. La de Bluetooth se basará en una aplicación movida en Android y la de Wifi en el propio navegador web existente en cada dispositivo que leerá la web suministrada por el servidor.

Así pues, se va a ir explicando los diferentes partes del trabajo según un orden que facilite el entendimiento.

4.1. Diseño hardware del Servidor.

El diseño hardware del sistema es el encargado de tener su "estructura física", donde correrá el programa y que permitirá la comunicación con los diferentes elementos del conjunto.

Ilustración 12: Jerarquía Hardware simplificada del Sistema.

Los archivos globales del hardware del sistema son:

- Wifi: Proyecto1_0.QPF
- Bluetooth: Bluethooth1_0.QPF

4.1.1. Top Hardware.

Prácticamente, en este caso, no tiene otra función que la de organizar el procesador NIOS del sistema y las entradas y salidas de éste hacia el exterior de la fpga y del exterior a éste.

Los archivos de “Verilog” son:

- Wifi: Proyecto1_0.v
- Bluetooth: Bluethooth1_0.v

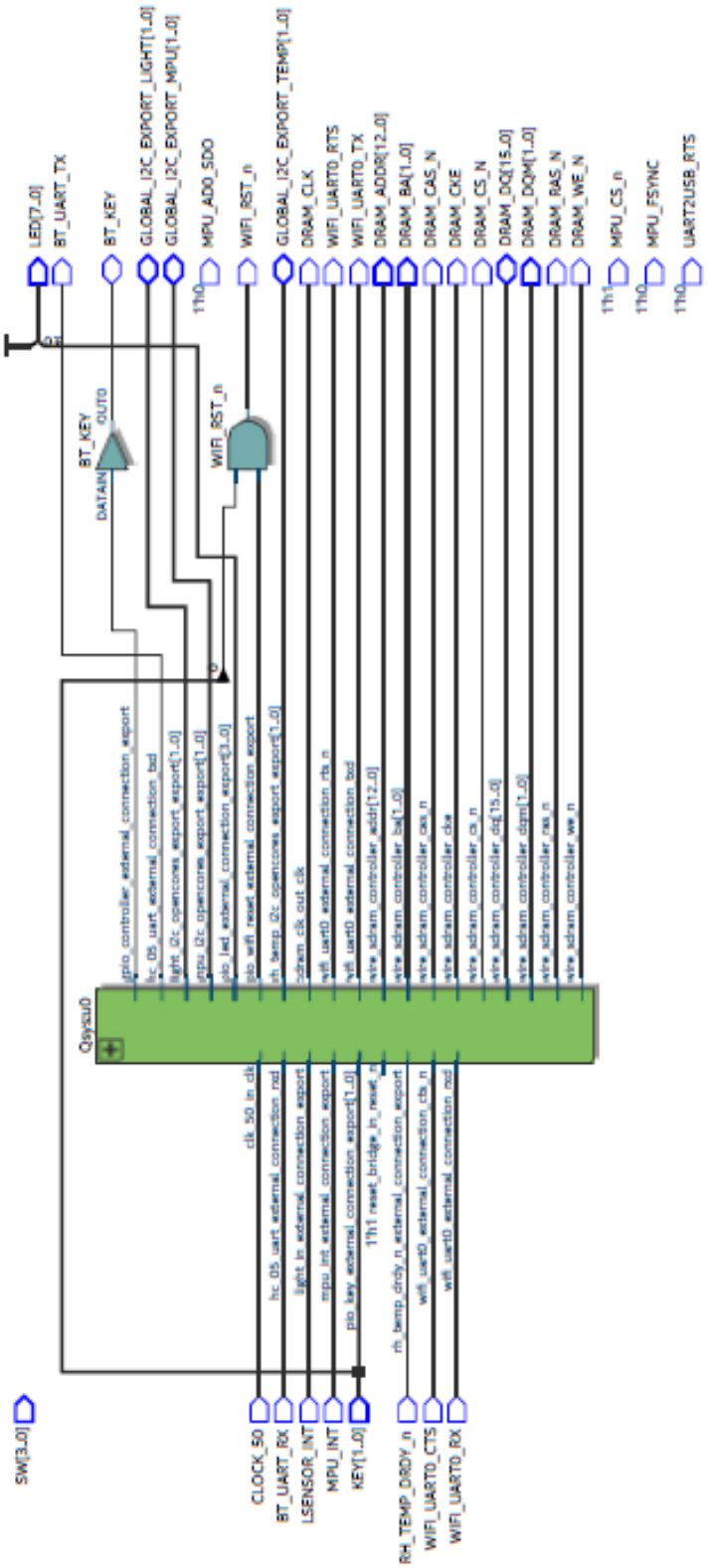


Ilustración 13: Top sistema Hardware

4.1.2. Hardware relacionado directamente con el procesador NIOS II.

Este es el hardware que engloba el micro soft. y a todos sus derivados, se ha diseñado mediante la herramienta “Qsys” de “Quartus Prime”. Para permitir una explicación clara del sistema se ha separado la exposición de los elementos en grupos con funcionalidades parecidas:

Bloque base:

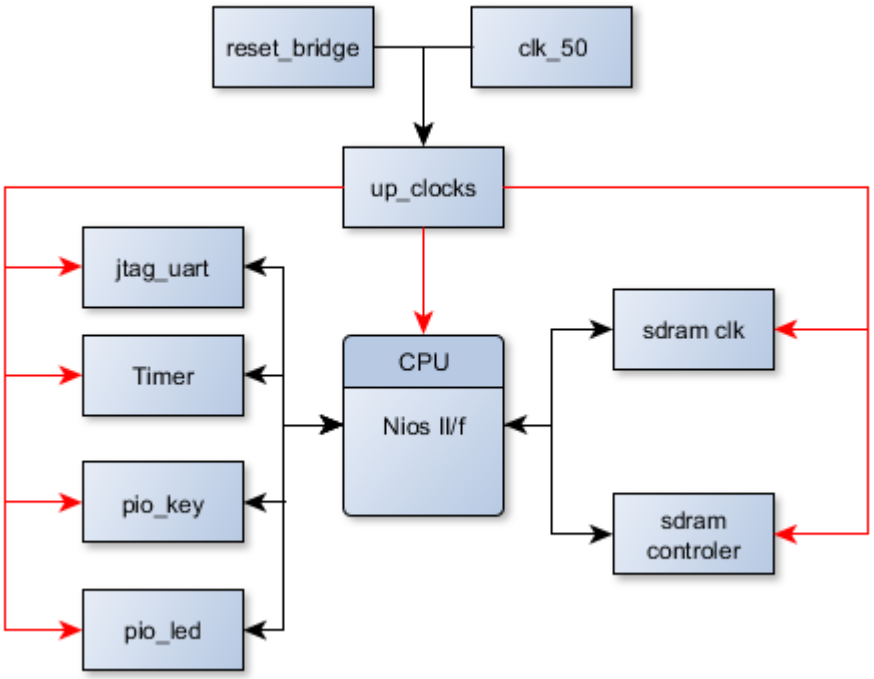


Ilustración 14: Bloque base sistema NIOSII.

Aquí podemos ver los componentes fundamentales para el correcto funcionamiento del sistema como:

- Up_clocks: Brinda la señal de reloj de 50Mhz a los elementos internos de la fpga.
- Jtag_uart: Permite la carga del programa y los procesos de depuración, entre otras tareas.
- Cpu: Es un microprocesador integrada de manera soft. en la Fpga de las siguientes características:

Tabla 5: Características NIOS II

Nios II Core: ○ Nios II/e ◉ Nios II/f

	Nios II/e	Nios II/f
Summary	**Resource-optimized 32-bit RISC**	**Performance-optimized 32-bit RISC**
Features	**JTAG Debug** **ECC RAM Protection**	**JTAG Debug** **Hardware Multiply/Divide** **Instruction/Data Caches** **Tightly-Coupled Masters** **ECC RAM Protection** **External Interrupt Controller** **Shadow Register Sets** **MPU** **MMU**
RAM Usage	2 + Options	2 + Options

- Sdram: Se ha decidido usar la sdram externa para el almacenamiento del programa por su mayor capacidad y por aprovechar los recursos del sistema.

Además de otros básicos:

- Timer: Brinda una temporización.
- Pio_key, pio_led: Son puertos paralelos ya preconfigurados para usar los interruptores y los led de la tarje DE0-nano.

Bloque sensores:

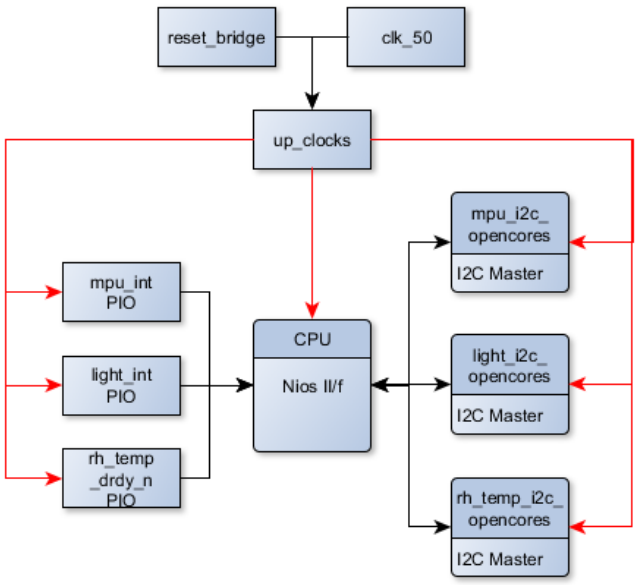


Ilustración 15: Bloque sensores sistema NIOSII.

Aquí podemos observar los módulos encargados de la comunicación con los sensores de la placa “RFS”. Esta comunicación se hace a través del estándar de

comunicación I2C usando una “IP” libre, además como se puede observar también hay unos puertos paralelos de entrada a la cpu.

Es interesante hacer referencia que se ha tenido que variar la IP original de I2C porque qsis no acepta varios “conduits” diferentes, sino que hay que agruparlos en un único en modo bus. Los archivos modificados son:

- i2c_opencores.v.
- i2c_opencores_hw.tcl

Bloque comunicaciones:

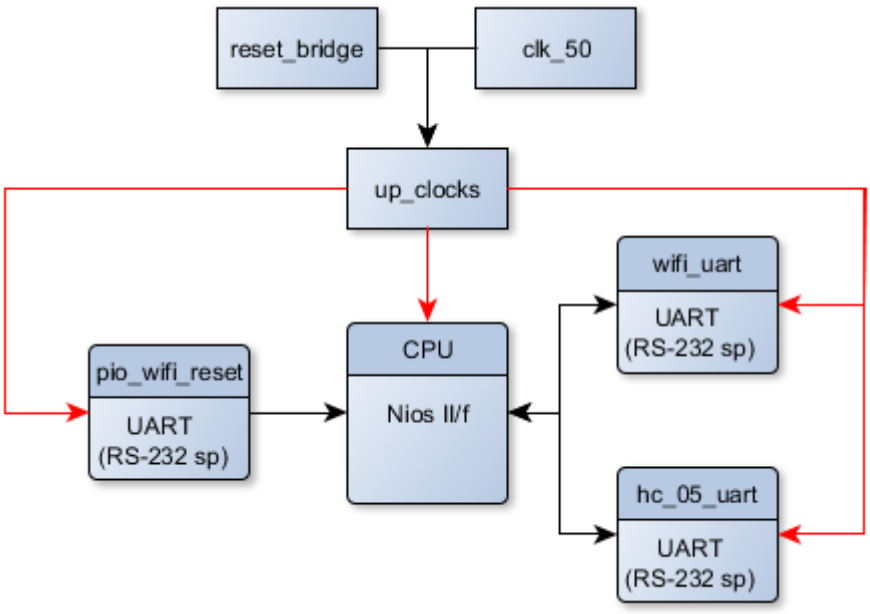


Ilustración 16: Bloque comunicaciones sistema NIOSII.

Aquí podemos observar los módulos que intervienen en la comunicación con el Wifi y el Bluetooth:

Wifi_uart: Uart con las siguientes características.

- Data bits: 8bits
- Stop bits: 1bit.
- Estados de sincronización (Segmentación) : 2.
- Baud rate: 115200 bps.
- Fijado el baud rate.

Hc_05_uart: Uart con las siguientes características.

- Data bits: 8bits
- Stop bits: 1bit.
- Estados de sincronización (Segmentación): 2.
- Baud rate: 38400 bps

- No fijado el baud rate.ç

Pio_wifi_reset: Es un puerto paralelo que permite el reset software del módulo del wifi.

4.1.3. Pines.

Cada salida/entrada que se encuentra en el top del archivo tiene referenciado un pin en la fpga. Estos pines pueden estar conectados a:

- La Ram de la placa DE0-nano.
- La placa de ampliación RFS a través del puerto de expansión GPIO-1.
- Los leds de la placa DE0-nano.
- Etc…

Todas las referencias de las salidas/entradas del archivo top se pueden encontrar en los archivos Excel:

- WiFiPines.excel
- BluetoothPines.excel

4.2. Diseño software.

Se han realizado dos programas diferentes dependiendo de la tecnología de transmisión de datos. Aunque básicamente la base del programa es la misma, se configura los módulos de comunicación y los sensores para después entrar en el bucle de servidor.

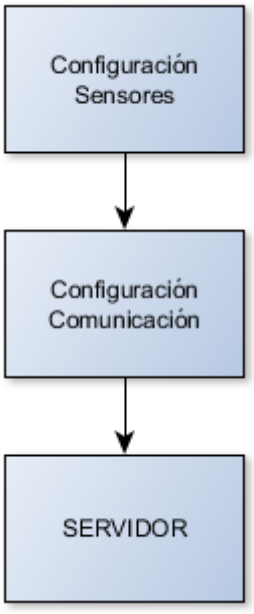


Ilustración 17: Diagrama de flujo software general.

4.2.1. Sensores.

Como ya sabemos tenemos tres chips diferentes que recogen en total 13 parámetros, para poder obtenerlos tenemos que usar los drivers proporcionados por Terasic que vamos a mencionar:

Inicialización y configuración:

- Temperatura (HDC1000):
    - RH_Temp_Init(RH_TEMP_I2C_OPENCORES_BASE)
    - RH_Temp_Sensor_Init()
- Sensor de luz:
    - Light_Init(LIGHT_I2C_OPENCORES_BASE)
    - Light_PowerSwitch(TRUE)
- 9-axis sensor(MPU9250)
    - MPU9250_Init(MPU_I2C_OPENCORES_BASE)
    - MPU9250_initialize()

Básicamente lo que realiza es la configuración e inicio de cada módulo I2C y una vez hecho esto la configuración de cada sensor.

Obtención de datos:

- HDC1000 (Temperatura):
    - Bool temp_response(float *fTemperature, float *fHumidity)
- Sensor de luz::
    - Bool luz_response(alt_u16 * light0, alt_u16 *light1)
- MPU9250(9-axis sensor)
    - Void motion_response(float * intMotion)

4.2.2. Bluetooth.

Como podemos observar en la imagen inferior el flujo del programa es el base comentado anteriormente. Lo que se diferencia, es la especificación en el bloque de comunicación, que se utiliza HC05 para el Bluetooth y el server que es específico para este.

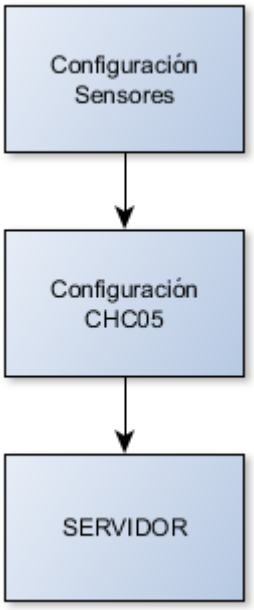


Ilustración 18: Diagrama de flujo Bluetooth base.

Así pues se va a explicar separadamente cada bloque:

Configuración HC05:

Es interesante comentar que los recursos dados por terasic para utilizar el módulo HC05 están en el lenguaje de c++. Es decir, se tiene una clase de HC05 con un conjunto de métodos y propiedades que nos servirán para inicializarlo, configurarlo y usarlo.

Así pues una vez hecho este pequeño inciso se va a comentar el procedimiento para su inicialización y configuración:

Como bien es sabido el módulo de HC05 se comunica a través de la UART del sistema para lo cual se deberá inicializar ambos:

- CUart Uart;
- CHC05 HC05;
- Uart.Open(HC_05_UART_NAME).

Una vez la Uart está operativa se tiene que entrar en el modo At, para ello se usa la siguiente función:

- SetConfig(true).

Cuando ya se está en modo AT se está en disposición de configurar el módulo para podernos conectar a un dispositivo con Bluetooth, pero antes de esto mencionemos la función de envió de datos de configuración del módulo CH05:

- bool CHC05::SetupSend(CUart Uart,char *cmd).

Configuración mediante comandos AT:

Comando: "AT+INIT\r\n"

Tabla 6: Comando AT+INIT:

29. Initialize the SPP profile lib

Command	Response	Parameter
AT+INIT	1. OK----success 2. FAIL----failure	None

+

Comando: "AT+UART=115200,0,0\r\n"

Tabla 7: Comando AT+UART.

13.Set/ Inquire- serial parameter

Command	Response	Parameter
AT+UART=<Param>,<Param2>,<Param3>	OK	Param1: baud rate(bits/s) The value (Decimal) should be one of the following: 4800 9600 19200 38400 57600 115200 23400 460800 921600 1382400 Param2:stop bit: 0----1 bit 1----2 bits Param3: parity bit
AT+ UART?	+ UART=<Param>,<Param2>,<Param3> OK	

Comando: HC05.SetupSend(Uart,"AT+ROLE=1\r\n")

Tabla 8: Comando AT+ROLE.

8. Set/ inquire module role

Command	Response	Parameter
AT+ROLE=<Param>	OK	Param: 0---- Slave role 1---- Master role 2---- Slave-Loop role Default: 0
AT+ ROLE?	+ ROLE:<Param> OK	

Comando: "AT+INQM=1,9,48\r\n"

Tabla 9: Comando AT+INQM.

11. Set/ inquire - Inquire access mode

Command	Response	Parameter
AT+INQM=<Param>, <Param2>,<Param3>	1. OK----success 2. FAIL----failure	Param: Inquire access mode 0----inquiry_mode_standard 1----inquiry_mode_rssi Param2: the maximum of Bluetooth devices response Param3:The maximum of limited inquiring time The range of limited time: 1~48 (Corresponding time:1.28s~61.44s) Default: 1, 1, 48
AT+ INQM?	+INQM:<Param>,<Param2>,<Param3> OK	

<u>Búsqueda de dispositivos mediante comandos AT:</u>

Comando: "AT+INQ\r\n".

Tabla 10: Comando AT+INQ.

30. Inquire Bluetooth device

Command	Response	Parameter
AT+INQ	+INQ: <Param1>,<Param2>,<Param3>, …… OK	Param1: Bluetooth address Param2: device type Param3: RSSI signal intensity

sprintf(szData,"AT+RNAME?%s\r\n",szDeviceList[i]);

HC05.GetNameList(Uart,szData);//Obtiene el nombre del dispositivo de Bluetooth.

Tabla 11: Comando AT+RNAME.

7. Get the remote Bluetooth device's name

Command	Response	Parameter
AT+RNAME?<Param1>	1. +NAME:<Param2> OK----success 2. FAIL----failure	Param1: Remote Bluetooth device address Param2: Remote Bluetooth device address

Conexión con el dispositivo seleccionado:

sprintf(szData,"AT+PAIR=%s,%d\r\n",szDeviceList[nChoice],30);

HC05.SetupSend(Uart,szData)

Tabla 12: Comando AT+PAIR.

32. Set pair

Command	Response	Parameter
AT+PAIR=<Param1>,<Param2>	1. OK----success 2. FAIL----failure	Param1: Bluetooth address of remote device Param2: limited time of connection (second)

sprintf(szData,"AT+BIND=%s\r\n",szDeviceList[nChoice]); Set/Inquire - bind Bluetooth address

HC05.SetupSend(Uart,szData); -Selecciona el deseado y se une.

Tabla 13: Comando AT+BIND.

15. Set/Inquire - bind Bluetooth address

Bluetooth address will show as this way: NAP: UAP:LAP(Hexadecimal)

Command	Response	Parameter
AT+BIND=<Param>	OK	Param----Bluetooth address needed to be bind Default address: 00:00:00:00:00:00
AT+ BIND?	+ BIND:<Param> OK	

Resumen:

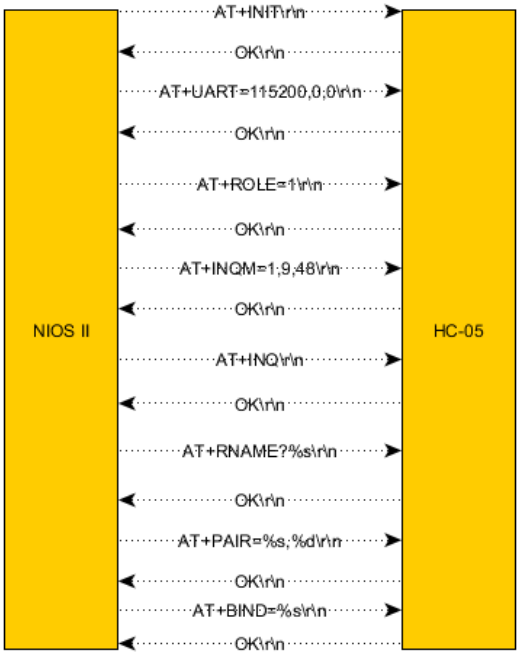


Ilustración 19: Conjunto comandos At Bluetooth.

Una vez ya se está conectado al dispositivo externo, se está en disposición de entrar en modo automático del servidor, para ello necesitamos salirnos del modo AT:

- SetConfig(false);

Servidor:

En el diagrama inferior podemos observar el flujo del bucle del servidor:

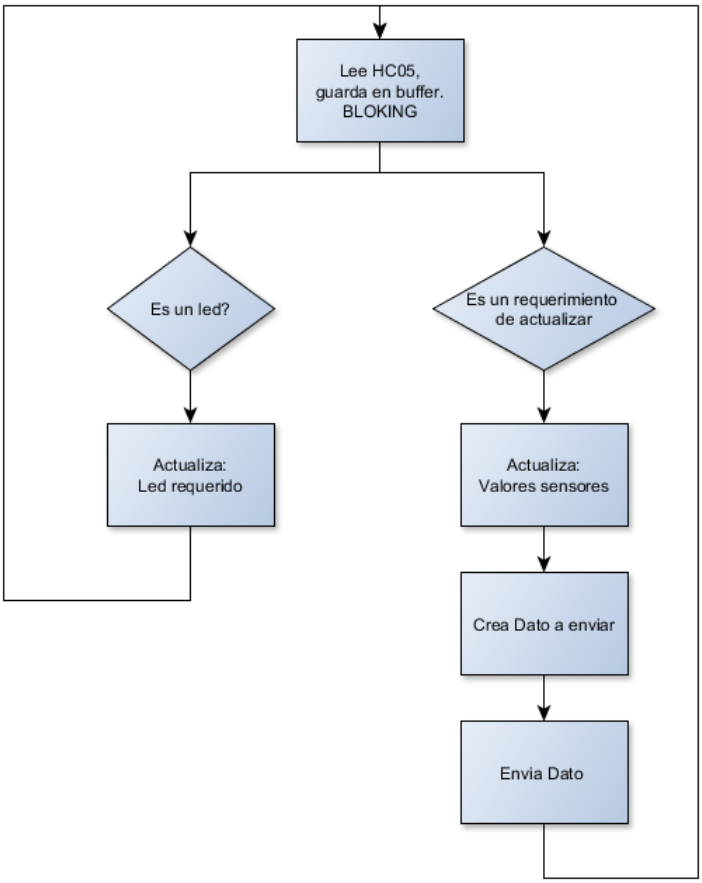


Ilustración 20: Diagrama de flujo del Servidor Bluetooth.

Básicamente lo que realiza es esperar hasta recibir una orden del dispositivo conectado y una vez recibida actuar en base a la orden. Solamente existen dos tipos de orden, una en modo de actuador con los Leds de la tarjeta y la otra que actualice y envíe los valores de los sensores.

Se va a comentar cada bloque:

Lee HC05:

- HC05.MessageRead(Uart, ReveiveData, sizeof(ReveiveData), &nReadLen), Simplemente lee la uart de manera bloqueante y una vez recibido algún dato lo almacena en ReceiveData.

Actualiza Led:

- led_response(led_num);

Actualiza valores sensores, hecho que ya se ha comentado anteriormente.

Crear Dato:

Se ha realizado una función específica para realizar esta función:

- void crear_dato(float * fTemperatura, float * fHumedad, alt_u16 * light0, alt_u16 *light1, float * intMotion2, bool bTemp, bool bLuz,char * puntTotal)

El siguiente diagrama de flujo dará una idea del funcionamiento de la función y del dato creado:

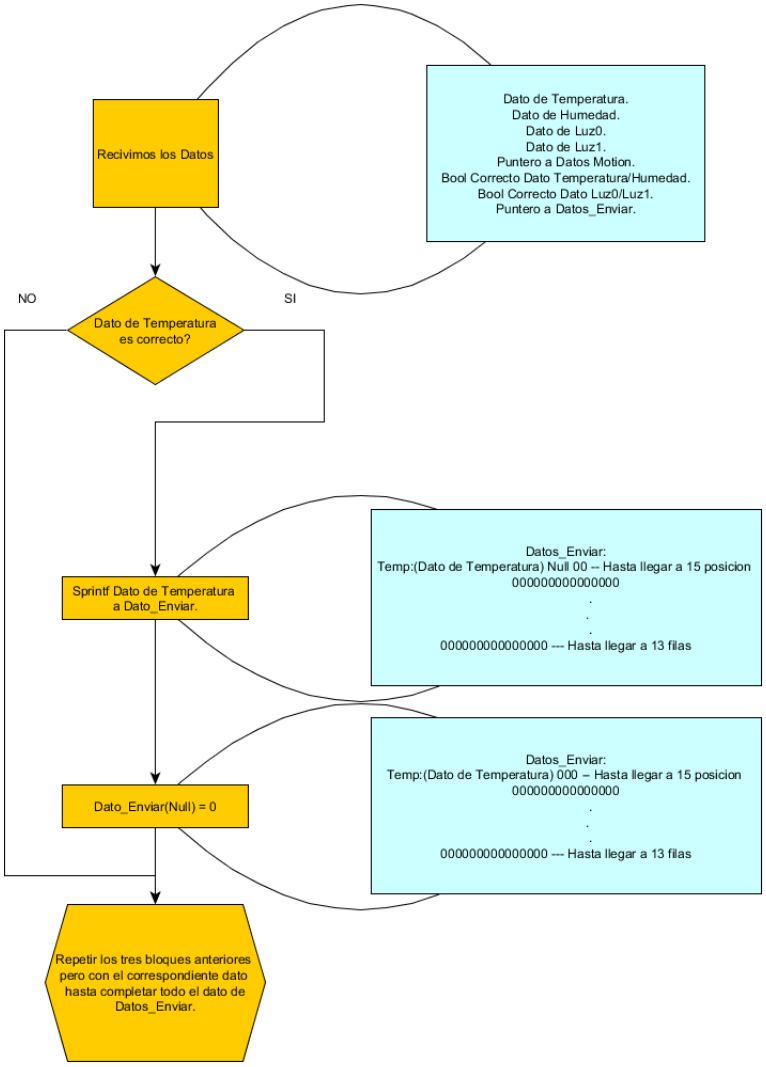


Ilustración 21: Diagrama flujo crear dato Servidor Bluetooth.

Envía valores de los sensores:

Una vez que se tiene el dato es el momento de enviarlo a través de la función:

- HC05.MessageSend(Uart,Datos_Enviar);

4.2.3. Wi-Fi.

Este es el caso del software con comunicación en base al modelo TCP/IP y como capa de aplicación el protocolo HTTP, ya que lo que se ha montado al lado del cliente es una página web.

Se puede observar en el diagrama de flujo inferior, que la metodología seguida para poner en marcha el servidor del Wi-Fi es igual a la del Bluetooth.

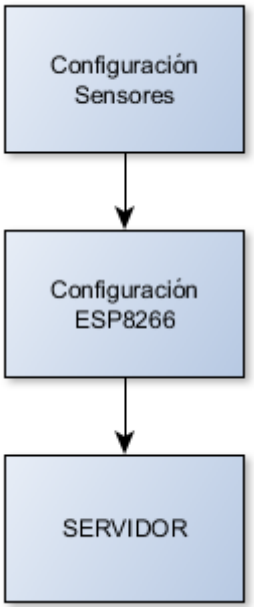


Ilustración 22: Diagrama de flujo Wi-Fi base

Configuración del ESP8266:

esp8266_file = fopen(esp8266_uart, "rw+"), modo ANSI C para inicializar la Uart, que se comunica con el módulo ESP8266.

```
 if (reset) {
      IOWR_ALTERA_AVALON_PIO_DATA(PIO_WIFI_RESET_BASE, 0);
      usleep(50);
      IOWR_ALTERA_AVALON_PIO_DATA(PIO_WIFI_RESET_BASE, 1);
      usleep(3 * 1000 * 1000);
      esp8266_dump_rx();
  }
```

Básicamente lo que hace es resetear el módulo de esp8266 y configurar el archivo. Una vez se ha inicializado éste, se configura el módulo para que sea un punto de acceso virtual:

"AT+CWSAP_CUR=\"Terasic_RFS\",\"1234567890\",5,3"

Tabla 14: Comando AT+CWSAP_CUR.

11. AT+CWSAP_CUR – Current config of softAP mode

AT+CWSAP_CUR – Set configuration of softAP mode, won't save to flash	
Example	AT+CWSAP_CUR="ESP8266", "1234567890", 5, 3
Command	**AT+CWSAP_CUR?**
Response	+CWSAP_CUR:<ssid>, <pwd>, <chl>, <ecn>, <max conn>, <ssid hidden>
Parameters	The same as below.
Command	**AT+CWSAP_CUR=<ssid>, <pwd>, <chl>, <ecn>[, <max conn>][, <ssid hidden>]**
Response	OK or ERROR // wrong parameter
Parameters	<ssid> string, ESP8266 softAP's SSID <pwd> string, range: 8 ~ 64 bytes ASCII <chl> channel ID <ecn> 0 : OPEN 2 : WPA_PSK 3 : WPA2_PSK 4 : WPA_WPA2_PSK [<max conn>], optional, default is 4 maximum count of stations that are allowed to connect to ESP8266 soft-AP range: [1, 4] [<ssid hidden>], optional, broadcast SSID by default 0 : broadcast SSID of ESP8266 soft-AP 1 : do not broadcast SSID of ESP8266 soft-AP
Notes	This command is only available when softAP is active. ESP8266 softAP does not support WEP. Configuration changes will NOT be stored in flash.

"AT+CWMODE_CUR=2"

Tabla 15: Comando AT+CWMODE_CUR.

2. AT+CWMODE_CUR – current WiFi mode

There are three Wi-Fi working modes: Station mode, softAP mode, and the co-existence of Station mode and softAP mode. This command is used to query the current Wi-Fi mode, or to set a desired Wi-Fi mode.

AT+CWMODE_CUR – Set WiFi mode(sta/AP/sta+AP), won't save to flash	
Example	AT+CWMODE_CUR=3
Command	**AT+CWMODE_CUR=?**
Response	+CWMODE_CUR:(value scope of <mode>) OK
Parameters	<mode> 1 : station mode 2 : softAP mode 3 : softAP + station mode
Command	**AT+CWMODE_CUR?**
Response	+CWMODE_CUR:<mode> OK
Parameters	<mode> 1 : station mode 2 : softAP mode 3 : softAP + station mode
Command	**AT+CWMODE_CUR=<mode>**
Response	OK
Parameters	<mode> 1 : station mode 2 : softAP mode 3 : softAP + station mode
Notes	Configuration changes will NOT be stored in flash.

Comando "AT+CWLIF":

Tabla 16: Comando AT+CWLIF.

13. AT+CWLIF – IP of stations

This command is used to get the IP of stations that are connected to ESP8266 softAP.

AT+ CWLIF- IP of stations which are connected to ESP8266 softAP	
Response	<IP addr>, <mac> OK
Parameters	<IP addr> IP address of stations which are connected to ESP8266 softAP <mac> MAC address of stations which are connected to ESP8266 softAP
Notes	This command cannot get static IP, it is only available if DHCP is enabled.

Comando AT+CIPMUX=1:

Tabla 17: Comando AT+CIPMUX.

13. AT+CIPMUX – Enable multiple connections

AT+ CIPMUX – Enable or disable multiple connections	
Example	AT+CIPMUX=1
Command	**AT+CIPMUX?**
Response	+CIPMUX:<mode> OK
Parameters	<mode> 0 : single connection 1 : multiple connection
Command	**AT+CIPMUX=<mode>**
Response	OK If already connected, returns Link is builded
Parameters	The same as above.
Notes	1. "AT+CIPMUX=1" can only be set when transparent transmission disabled ("AT+CIPMODE=0") 2. This mode can only be changed after all connections are disconnected. 3. If TCP server is running, it must be deleted before single connection mode can be activated.

"AT+CIPSERVER=1,80"

Tabla 18: Comando AT+CIPSERVER.

14. AT+CIPSERVER – Configure as TCP server

Server monitor will automatically be created when Server is created. When a client is connected to the server, it will take up one connection and be assigned an ID.

AT+CIPSERVER - Configure as TCP server	
Example	AT+CIPMUX=1 AT+CIPSERVER=1,1001
Command	**AT+CIPSERVER=<mode>[, <port>]**
Response	OK
Parameters	<mode> 0 : Delete server 1 : Create server <port> port number, default is 333
Notes	Server can only be created when AT+CIPMUX=1

Una vez hay conexión tcp/ip, estamos listos para enviar o recibir datos:

- Los datos se recibirán a través del comando +IPD:

Tabla 19: Comando +IPD.

21. +IPD – Receive network data

+IPD - Receive network data	
Single connection	**(+CIPMUX=0)** **+IPD, <len>[, <remote IP>, <remote port>]:<data>**
Multiple connection	**(+CIPMUX=1)** **+IPD, <ID>, <len>[, <remote IP>, <remote port>]:<data>**
Parameters	<remote IP> remote IP, enabled by command "AT+CIPDINFO=1" <remote port> remote port, enabled by command "AT+CIPDINFO=1" <ID> ID number of connection <len> data length <data> data received
Notes	When the module receives network data, it will send the data through the serial port using +IPD command.

Resumen:

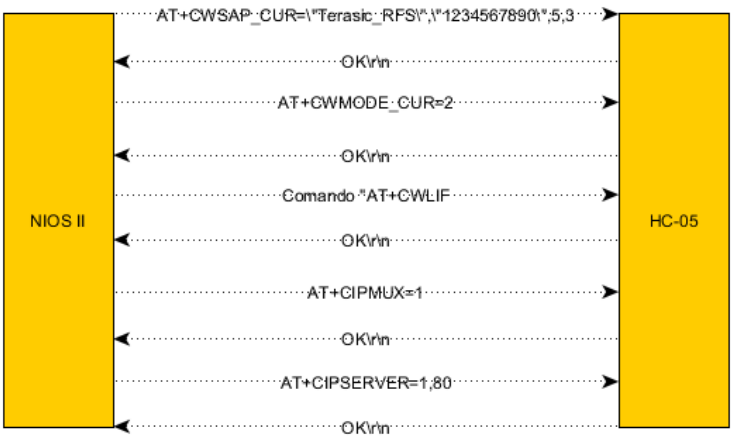


Ilustración 23: Conjunto comandos At Wi-Fi.

Servidor:

Una vez conectados el cliente al servidor a través de la conexión TCP/IP, se puede activar la comunicación entre ellos:

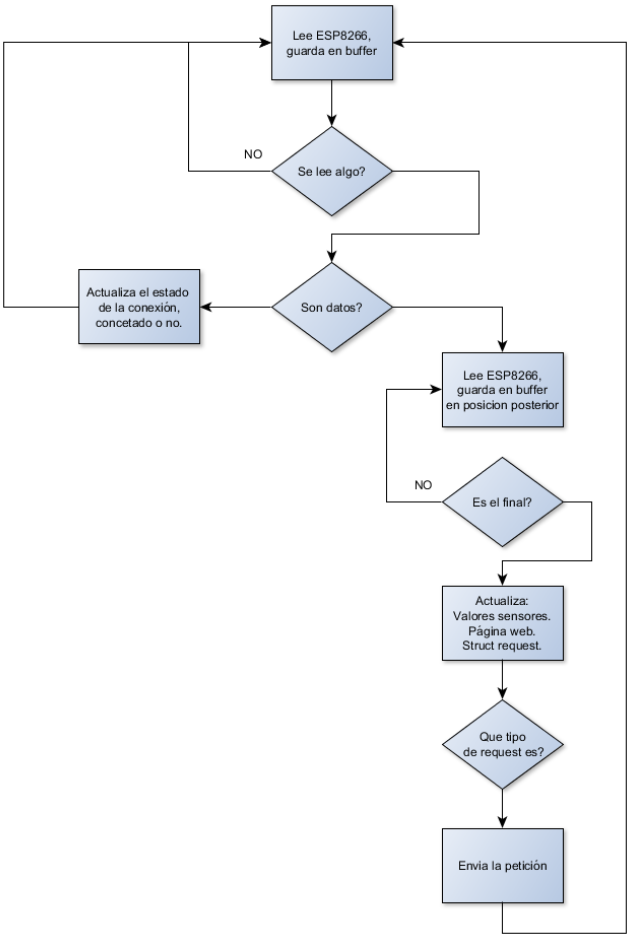


Ilustración 24: Diagrama de flujo del Servidor Wi-Fi.

El funcionamiento es parecido al caso del Bluetooth, ya que el sistema está a la espera de recibir algún tipo de orden observando constantemente (polling) al archivo que hace la función de periférico. Una vez recibe ese dato lo analiza para saber qué tipo de orden es y que dato se debe de enviar, las posibles parejas de orden/dato son las siguientes:

Tabla 20: Tipo de datos de la comunicación Wi-FI.

Resource type	Response		
	Status Code	Response Header	Body Message
/	200 OK	Content-Type: text/html	web_src/index.html
/favicon.ico	200 OK	Content-Type: text/plain	web_src/favicon.ico
/Logo_Terasic.jpg	302 Found	Content-Type: image/jpeg	web_src/Logo_Terasic.jpg
/led	302 Found	Location:/	---
/actualizar	302 Found	Location:/	---
/parar	302 Found	Location:/	---
Other	404 Not Found	Content-Type: text/html	web_src/404.html

Es decir, son requerimientos de la página web, ya sea la página web en html, una imagen o otros, y siempre mediante el estándar Http.

Se va a comentar algunos bloques para entrar más en detalle:

Lee ESP8266:

- fgets(buffer8266, sizeof(buffer8266), esp8266_file) ,la función utilizada sigue con el modo de ANSI C leyendo del archivo esp8266_file y almacenando el resultado, en el caso de que haya algo nuevo, en el buffer.

Actualización:

struct request:

```
struct http_request {
    enum http_methods_enum {
        GET, POST, OTHER
    } http_methods;
    char *path;
    bool connected;
    int id;
};
```

Es una estructura que recoge las características de la petición del cliente, como puede ser el método de la petición (GET, POST, OTHER), el path o dato que corresponde en resource de la Tabla 18 y el resto que tiene que ver con la conexión del cliente.

Actualización html:

Se ha creado una función específica para esta tarea:

- dinamic_html(&fTemperature2,&fHumidity2,&light0,&light1,intMotion,&request[id]),

La necesidad de esta función radica en que se debe pasar o enviar la página web con los nuevos parámetros de los sensores y no se dispone de un servidor dinámico, que por ejemplo tenga integrado el reconocimiento de PHP, y que de dinamismo al lado de éste. Así esta función la da, aprovechamos que realmente para el módulo Wi-Fi no es una página web, sino una secuencia de caracteres que es posible de modificar de la siguiente manera:

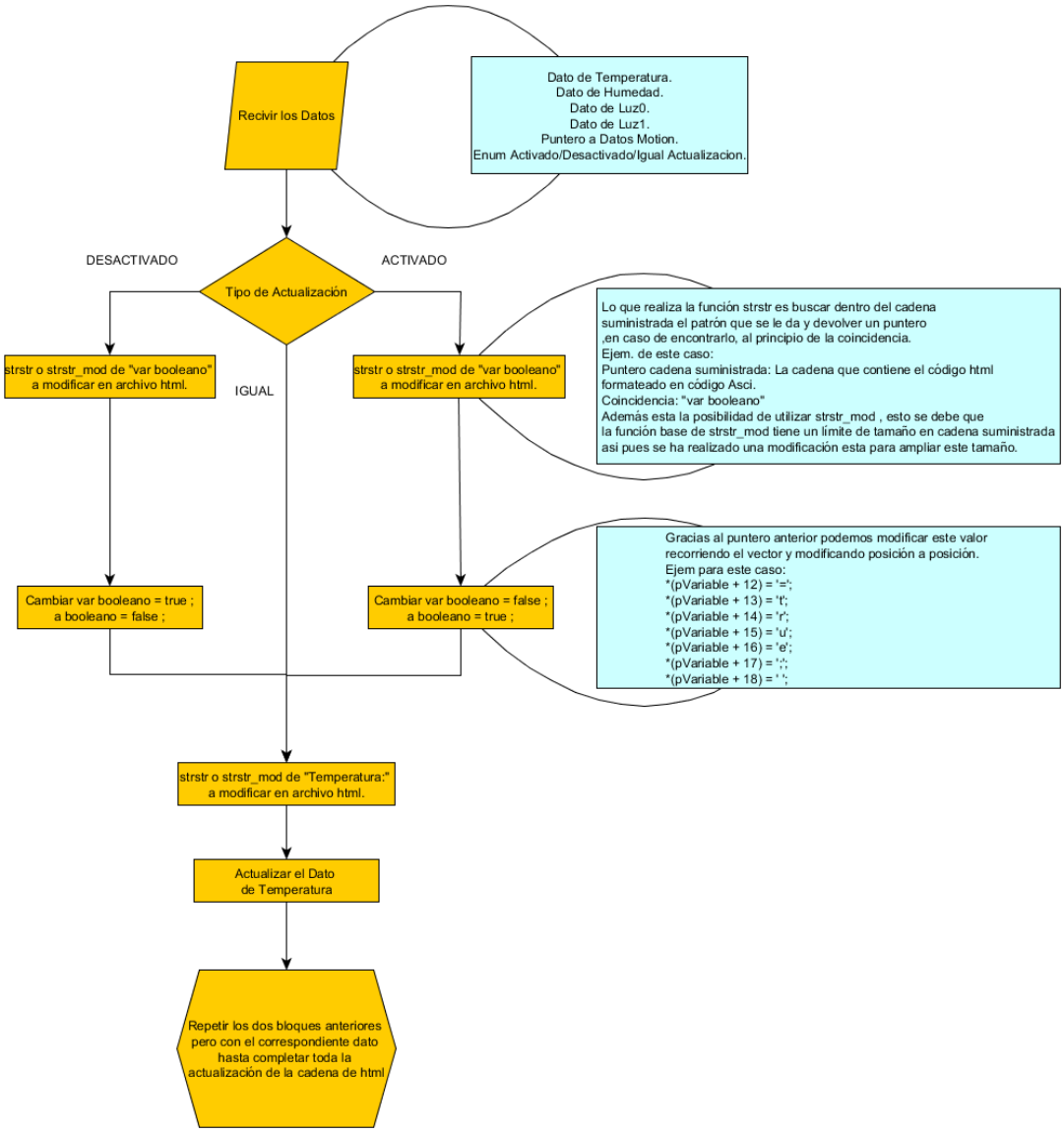


Ilustración 25: Modificación html dinámicamente.

Otro hecho interesante de comentar es como realiza el volcado de código fuente en html o de una imagen a una cadena hexadecimal. Esto se realiza con el comando xxd y con una compilación a propósito. Así pues, antes de compilar el proyecto en su globalidad, si se ha modificado el archivo de Html o cualquier recurso de la web, es necesario actualizarlo mediante este método.

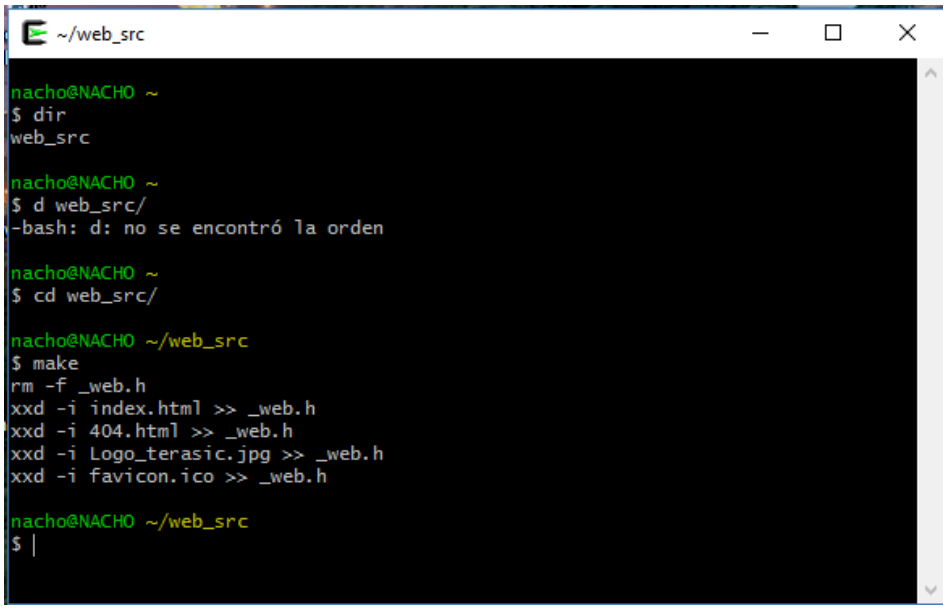
- 

Ilustración 26: Ejemplo volcado a archivo a caracteres en hexadecimal.

4.3. Cliente.

Como cliente también se tiene dos tipos diferentes de clientes dependiendo del servidor al que pertenezca.

4.3.1. Cliente Bluetooth.

El cliente Bluetooth está basado en una aplicación móvil proporcionada por Terasic que se ha modificado para adaptarla a las necesidades del trabajo. Está diseñado para un móvil con SO android que tenga comunicación por Bluetooth.

Los archivos base se encuentran dentro del archivo:

- RFS_v.1.2.0_SystemCD.

Los archivos modificados se encuentran dentro del archivo:

- Android_Project_mod.

La interfaz gráfica de la app es la siguiente:

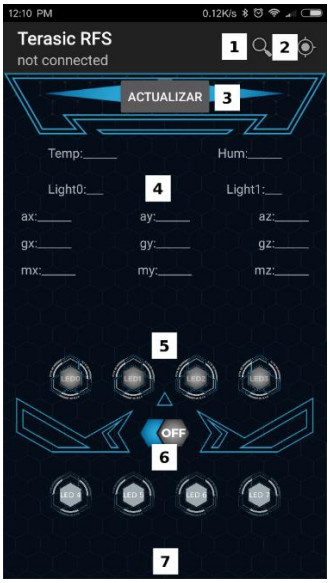


Ilustración 27: Interfaz gráfica cliente Bluetooth.

La funcionalidad de cada parte es la siguiente:

1: Es un buscador de Bluetooth.

2: Permite la conexión a Bluetooth.

3: Es el botón que inicia la actualización automática de los valores de los sensores o que la detiene en el caso que se pulse por segunda vez.

4: Son los parámetros de los sensores.

5: Son los actuadores unitarios de los Leds.

6: Es el actuador que tiene efecto sobre todos los Leds al unísono.

7: Lugar donde se muestra los mensajes enviados y recibidos.

Además de modificar la interfaz gráfica, se ha tenido que adaptar el programa para que envíe la petición de actualización de los datos de los sensores de manera síncrona.

Envío síncrono de la petición de actualización:

El cliente es el encargado de cada T tiempo enviar una petición de actualización de los datos que dispone al Servidor. Como la app está basada en Android, el cual es un sistema operativo que funciona mediante tareas, se hace necesaria la sincronización de éstas para que no interfieran entre sí y desincronicen la petición de actualización.

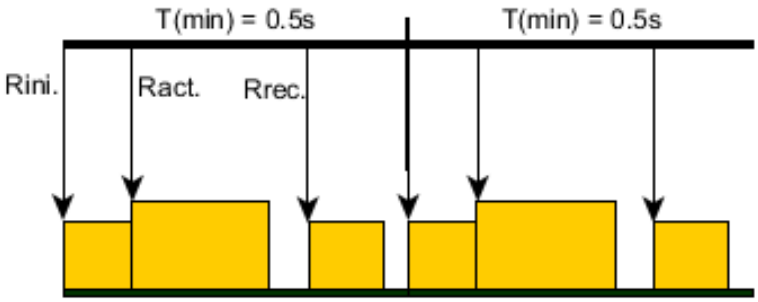


Ilustración 28: Opción 1ª tareas Android.

- Rini.: Inicio de envío de código al servidor.
- Ract.: Inicio de actualización datos en la interfaz gráfica, es la de mayor prioridad.
- Rrec.: Recibo de la información del servidor y tratamiento de esta.

Es decir, esperamos a enviar la petición de datos al servidor para empezar a actualizar la interfaz gráfica con el dato obtenido de una petición anterior. Esta manera parece errónea pero le da al usuario una impresión de fluidez y el dato real se trataría después de Rrec. de manera correcta.

En caso de realizar la actualización de la interfaz después de obtener los datos del servidor, pasaría lo siguiente:

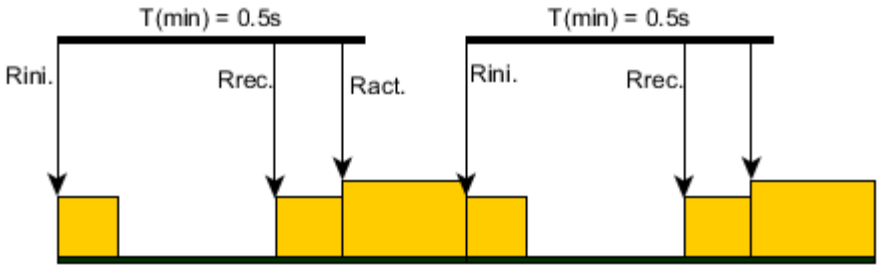


Ilustración 29: Opción 2ª tareas Android.

Es decir, la actualización se haría con el dato pedido pero no habría una sincronía, lo que haría sería ir a saltos en la percepción del usuario, ya que el sistema operativo intentaría solventar este problema adelantando otras peticiones de datos. Este hecho se verá más claro en el apartado de pruebas.

Una vez explicado la necesidad de realizar la sincronización, vamos a desarrollar mediante diagramas de flujo como se ha hecho:

Hay que tener en cuenta que solo se está explicando una pequeña parte de la aplicación, que es la que interviene directamente en el problema citado, aun así es interesante saber que la aplicación tiene al menos dos procesos trabajando:

- Un Activity, encargado de la interfaz gráfica.
- Un Service, encargado de manejar el Bluetooth.

Así pues, el primer diagrama tiene que ver con el botón que indicara al Timer si debe enviar o no la petición de actualización (Rini) y que estará dentro del Activity:

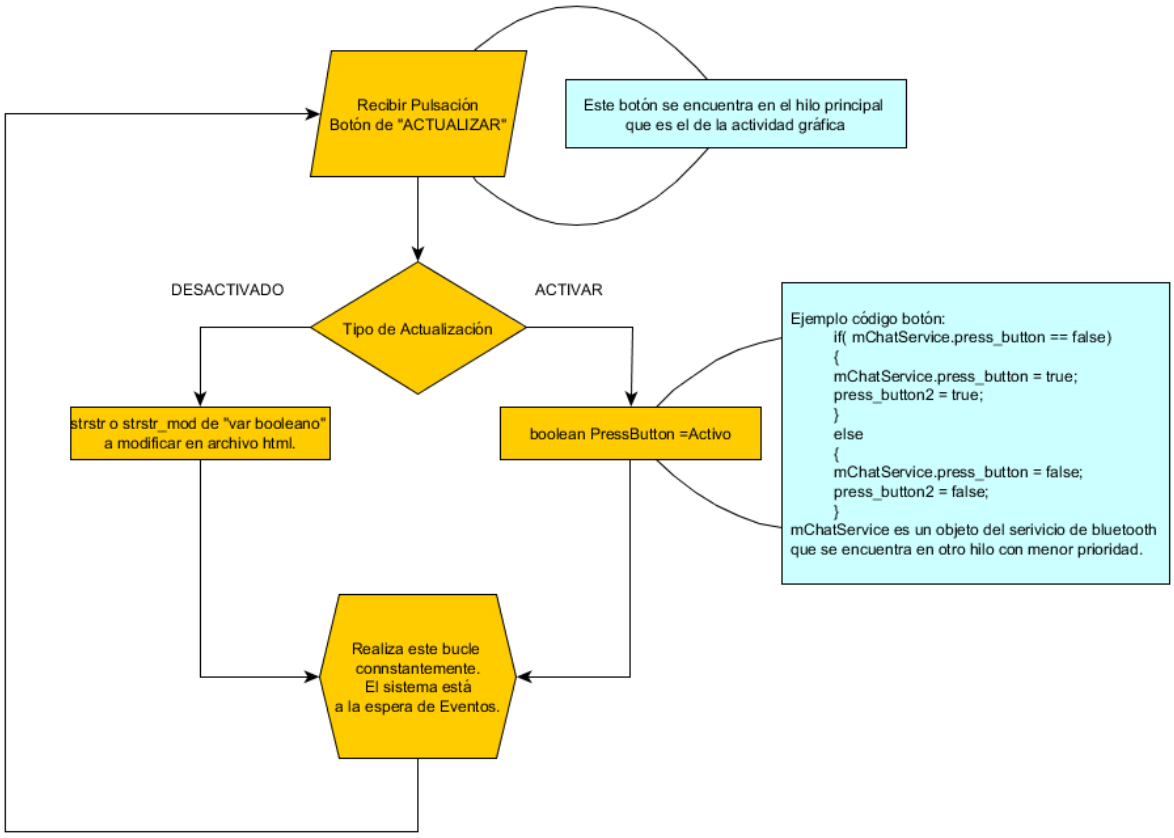


Ilustración 30: Diagrama flujo evento botón en el cliente Bluetooth.

A su vez estará el Timer en el Service comprobando cíclicamente si enviar o no las peticiones necesarias.

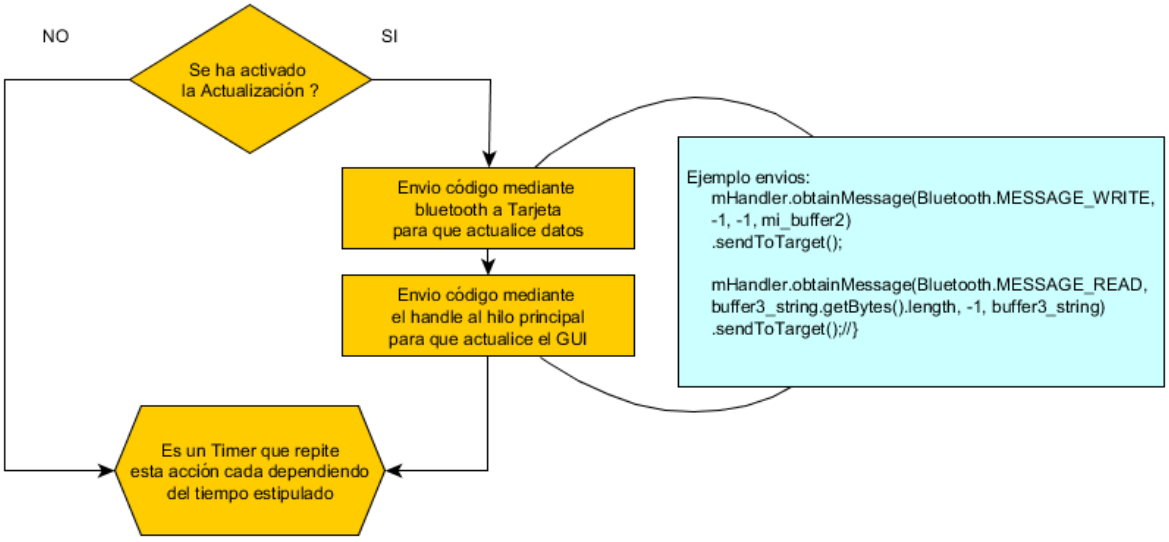


Ilustración 31: Diagrama del Timer en el cliente Bluetooth.

Como se puede observar es esta tarea la que envía la petición de actualizar la interfaz de usuario después de la de actualización de los datos, lo que asegura que no la solape.

Así quedarían dos partes por desarrollar:

1. Por una parte la actualización de la Gui, una vez recibida la petición, que se haría en el manejador de la Activity. Esta clase/objeto es la encargada de recibir y enviar los datos al otro proceso Service.
2. Recepción por parte del Service de los datos enviados y almacenamiento en variables locales que utilizara la Activity para actualizar los datos.

4.3.2. Cliente Wi-Fi.

El cliente wifi, está basado en la página web corriendo en el navegador web, el cual es una aplicación ya dado por sistemas operativos como android, windows etc...

La interfaz gráfica es la siguiente:

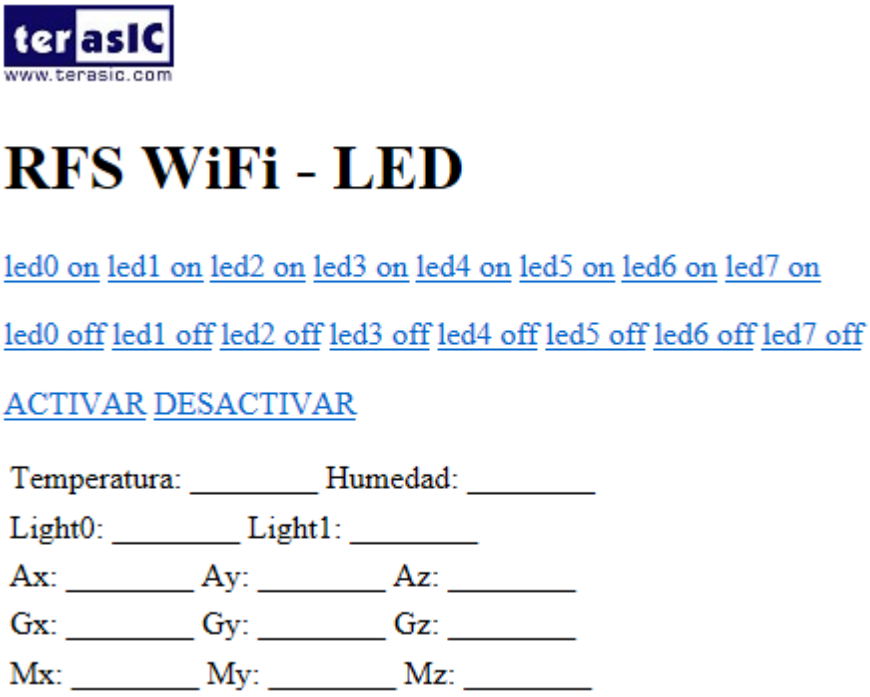


Ilustración 32: Pagina Web.

Descripción:

Como se puede observar la página web contiene dos partes, en la parte superior la de activación/desactivación de los Leds o los Datos de los Sensores. En la parte inferior donde se muestran los Datos de los Sensores actualizados.

Página web en html:

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8">
    <title>RFS WoFi - LED</title>
    <script>

    var booleano=false;
    </script>

</head>


<body>
    <img src="Logo_Terasic.jpg" alt="Terasic Logo" height="38px" width="85px">
    <h1>RFS WiFi - LED</h1>
    <p>
    <a href="led/on/0">led0 on</a>
    <a href="led/on/1">led1 on</a>
    <a href="led/on/2">led2 on</a>
    <a href="led/on/3">led3 on</a>
    <a href="led/on/4">led4 on</a>
    <a href="led/on/5">led5 on</a>
    <a href="led/on/6">led6 on</a>
    <a href="led/on/7">led7 on</a>
    </p>
    <p>
    <a href="led/off/0">led0 off</a>
    <a href="led/off/1">led1 off</a>
    <a href="led/off/2">led2 off</a>
    <a href="led/off/3">led3 off</a>
    <a href="led/off/4">led4 off</a>
    <a href="led/off/5">led5 off</a>
    <a href="led/off/6">led6 off</a>
    <a href="led/off/7">led7 off</a>
    </p>
    <p>
    <a href="activar">ACTIVAR</a>
    <a href="desactivar">DESACTIVAR</a>
    </p>
    <p>
    <table>
        <tr>
            <td>Temperatura: ________  </td>
            <td>Humedad: ________ </td>
        </tr>
    </table>
    <table>
        <tr>
            <td>Light0: ________  </td>
            <td>Light1: ________  </td>
        </tr>
    </table>
    <table>
        <tr>
            <td>Ax: ________  </td>
            <td>Ay: ________  </td>
            <td>Az: ________  </td>
        </tr>
    </table>
    <table>
        <tr>
```

```
63              <td>Gx: _________  </td>
64              <td>Gy: _________  </td>
65              <td>Gz: _________  </td>
66          </tr>
67      </table>
68      <table>
69          <tr>
70              <td>Mx: _________  </td>
71              <td>My: _________  </td>
72              <td>Mz: _________  </td>
73          </tr>
74      </table>
75      </p>
76 <script>
77
78 if (booleano == true) setTimeout('document.location.reload()',10000);
79
80 </script>
81
82 </body>
83 </html>
```

Ilustración 33: Página Web en Html.

Como se puede observar además de estar el lenguaje de marca HTML existe código ejecutable en JavaScript para permitir la actualización de la web obteniendo los Datos de los Sensores.

A continuación se va a mostrar el funcionamiento de manera genérica del Cliente Wi-Fi:

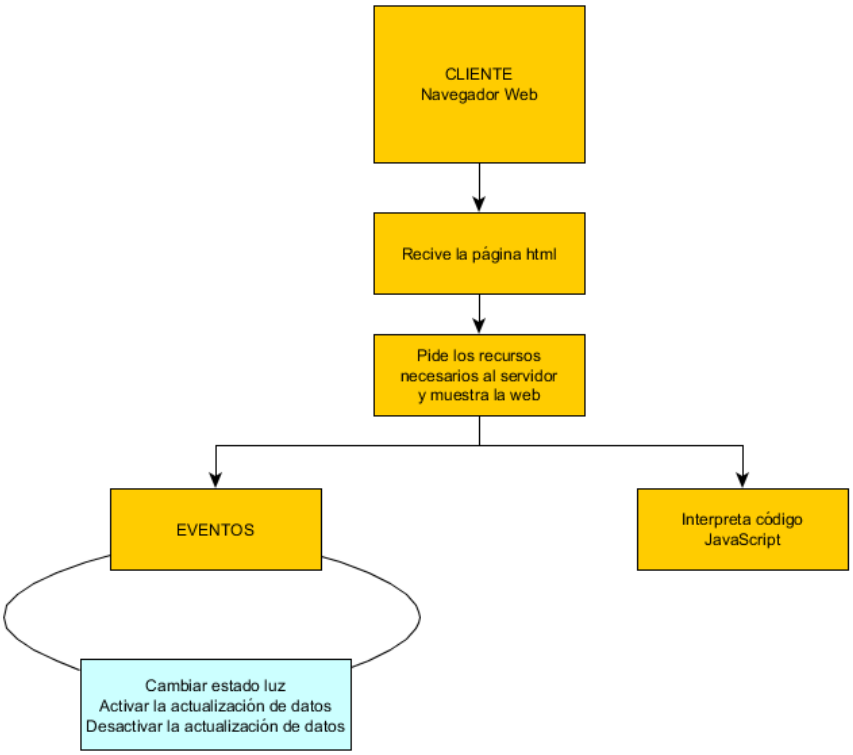


Ilustración 34: Diagrama flujo modo de funcionamiento Cliente Wi-Fi.

Así sería el funcionamiento del navegador web, en el caso que se activará algún tipo de evento, el navegador enviaría una solicitud siguiendo el protocolo HTTP al servidor.

Por ejemplo si le activáramos el Led0, la respuesta del navegador sería la siguiente:

- +IPD, 0, 449:GET /led/on/0 HTTP/1.1, esto sería lo que recibiría el Servidor.

En el caso de que en lugar de activar un Led, activáramos la actualización de los Datos de los Sensores, sucedería lo siguiente:

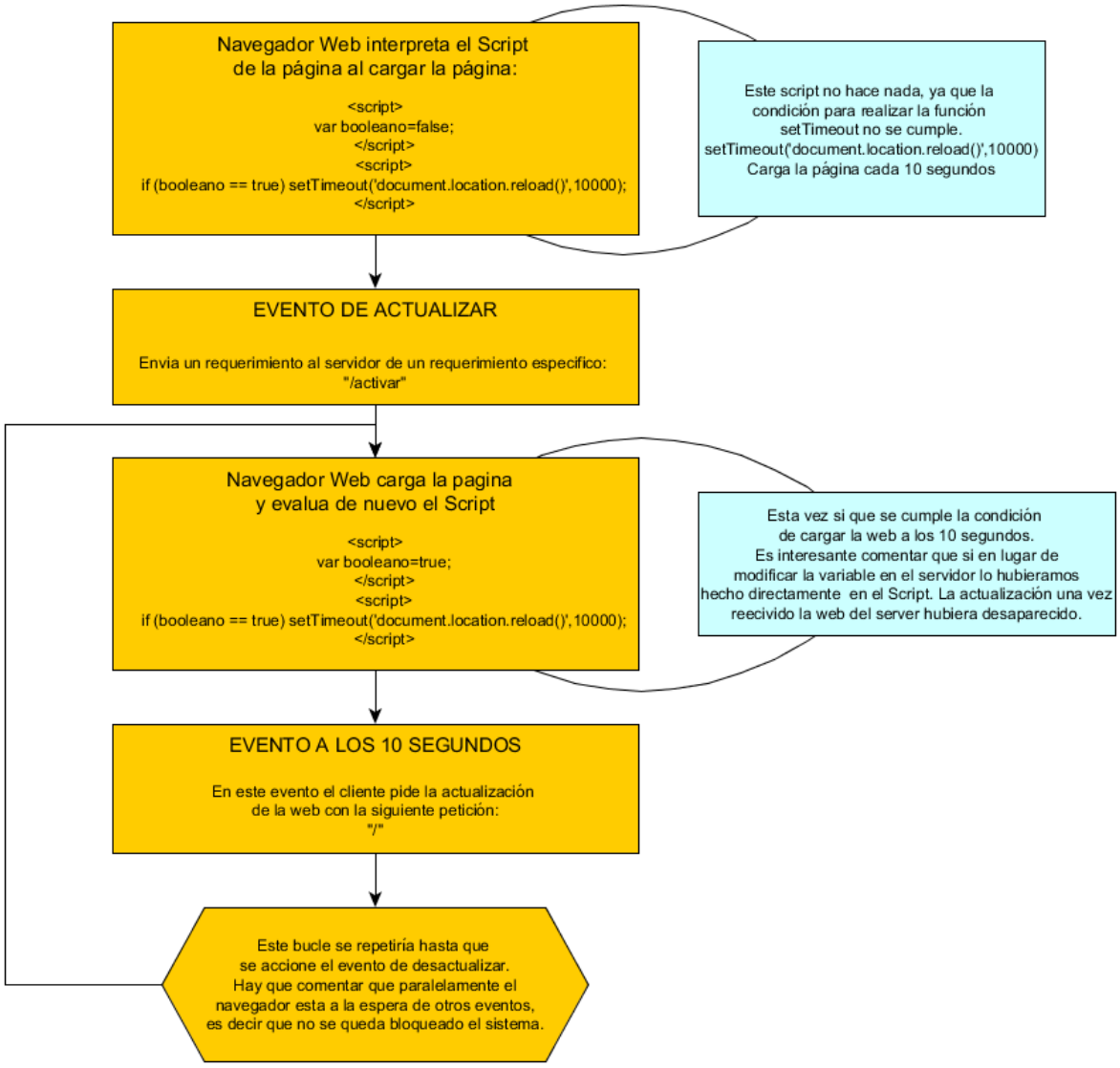


Ilustración 35: Diagrama flujo modo de funcionamiento de un caso específico Cliente Wi-Fi.

Es decir se crearía un Timer “ficticio” que actualizaría el sistema cada T segundos.

5. Resultados y pruebas.

Los resultados se medirán a través de una función que captura los ticks actuales en el momento de su uso. De esta manera se puede obtener de una manera un poco bruta y sin una gran precisión, ya que hay 8 ticks por segundo, el momento de recepción de la orden de envío de actualización de los Datos de los Sensores o de la respuesta del Servidor.

La estructura es la siguiente:

```
#ifdef DEBUG_BLUETOOTH
start_time = alt_nticks();
total_time = ((start_time) * 1000) /
alt_ticks_per_second() ;
printf("Tiempo absoluto en que se encuentra el sistema en ese momento %d ms\n\n\n", total_time);
#endif
```

5.1. Bluetooth comprobación.

Aquí se puede observar el momento de recepción de la orden de actualización enviada por el Cliente Bluetooth y el momento de respuesta del sistema.

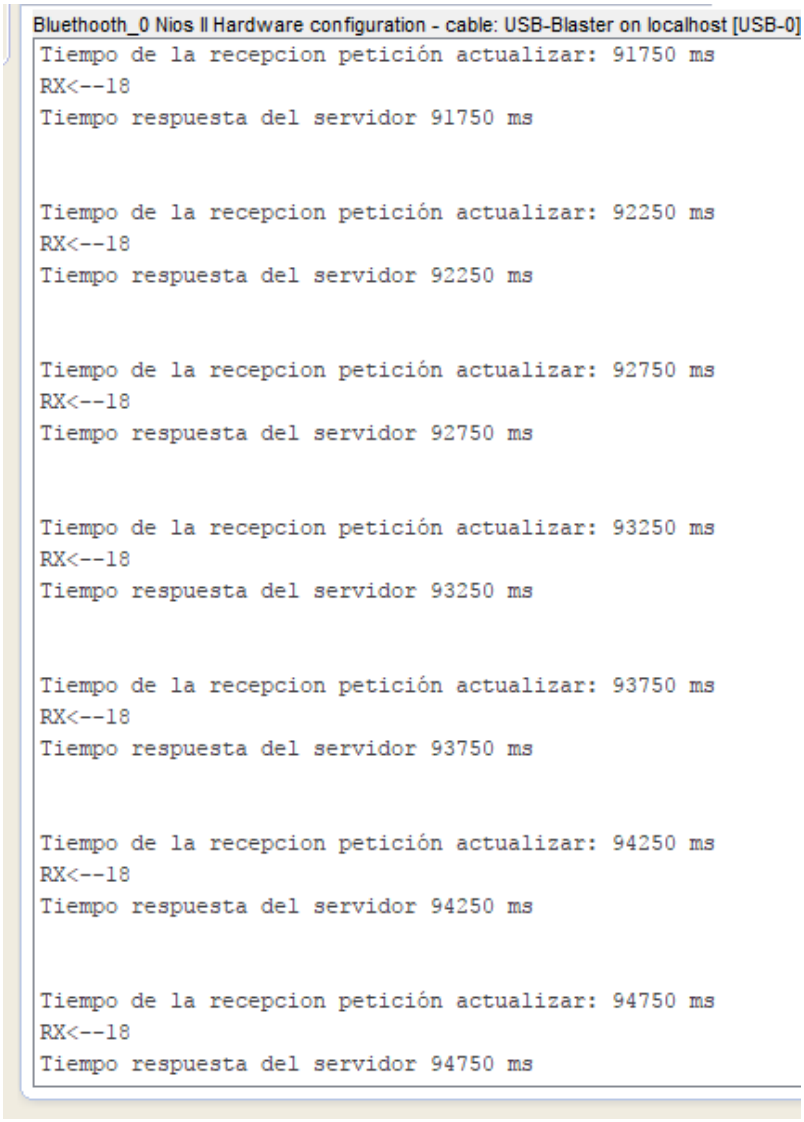


Ilustración 36: Comprobación sincronismo Bluetooth.

Como se puede observar el sistema avanza cíclicamente cada 0.5 segundos, e incluso se observa que todo el tratamiento no consume tiempo. Obviamente, sí que lo consume solo que la precisión del sistema no es suficiente para verlo.

Ahora supongamos que procedemos a realizar la actualización de la Interfaz Gráfica nada más recibir los datos del Servidor. Sucedería lo siguiente:

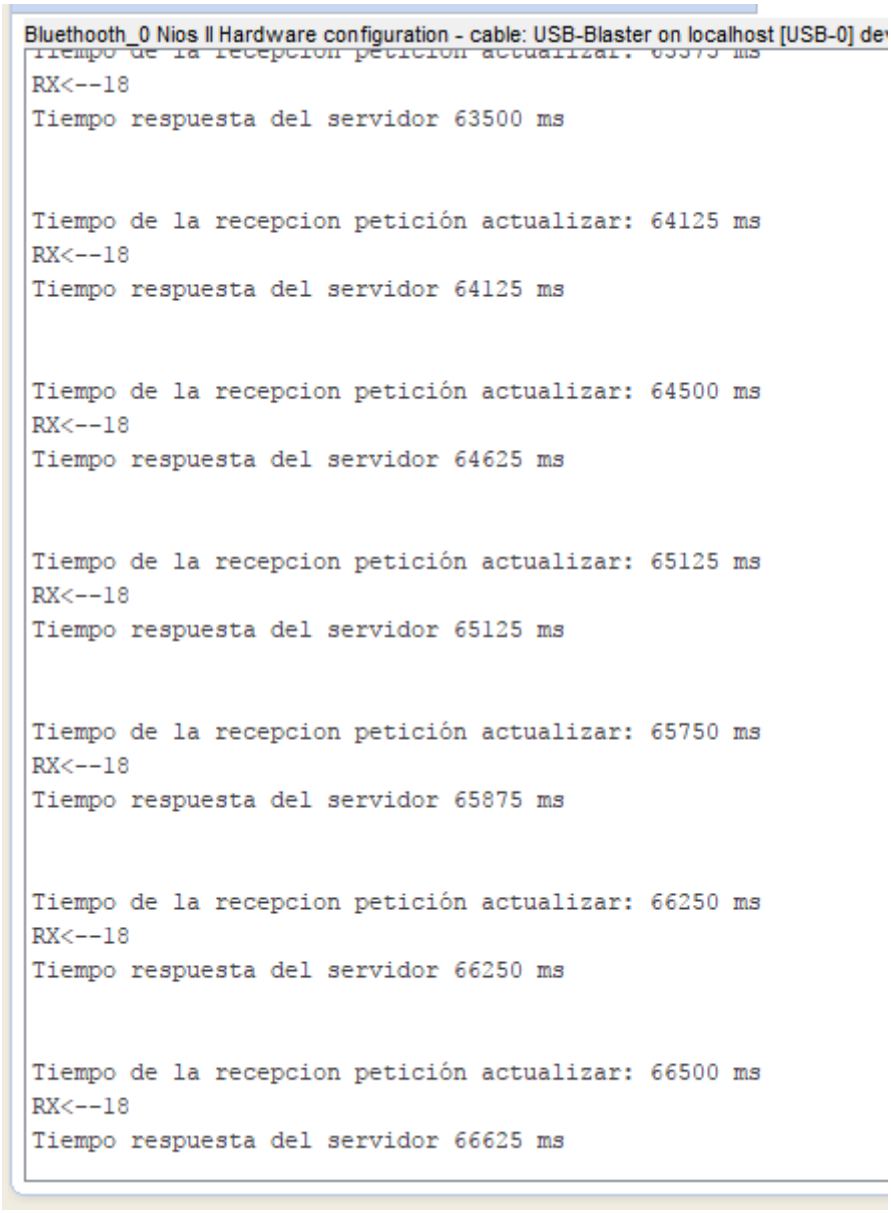


Ilustración 37: Comprobación no sincronismo Bluetooth.

Como se observa hay una pequeña desincronización, algunas veces tarda más de tiempo estipulado otras menos. Es decir, no es un sistema correcto ya que provocaría saltos en la percepción visual y en la recepción de los datos.

5.2. Resultados Bluetooth.

- Sincronía: Aceptable.
- Ancho Banda: 3.12 Kb/s.

Cálculo del ancho banda perfecto sin contabilizar:

- Tiempos del microprocesador NIOS II ni del sistema cliente.
- Tiempos transmisión a través del Wi-Fi.
- Obtención de los datos de los sensores de manera secuencial.

Sabemos que tenemos las siguientes características:

- Ancho Banda Módulo Bluetooth: 2 Mb/s – 3 Mb/s , realmente UART a 115.2 Kbits/s
- Ambient Light Sensor (APDS-9301): 400 Kb/s, pero tiene que obtener 16 bits, se podría decir que el tiempo de respuesta es 1/((400*10^3) / 16). → 40µs.
- Temperature and humidity sensor HDC1000, 400 Kb/s, pero tiene que obtener 14 bits, se podría decir que el tiempo de respuesta es 1/((400*10^3) / 14). → 35µs.
- 9-axissensor –accelerometer, magnetometer, and gyroscope (MPU-9250), pero tiene que obtener 16x9 bits, se podría decir que el tiempo de respuesta es 1/((400*10^3) / (16x9)). → 360µs.

Es decir que en el mejor de los casos y obteniendo los Datos de los Sensores de manera secuencial, se tardaría en obtener éstos:

$$T_{Sensorización} = \sum (T_{SensorLuz} + T_{SensorTemp.} + T_{SensorMot.}) = 40\mu s + 35\mu s + 360\mu s = 435\mu s$$

Serían 435 μs en obtener 174 bits que a una Frecuencia marcada por la Uart de 115.2 Kbits/s tardaría otros 174/(115.2*10^3) = 1.51ms segundos más.

Así el tiempo en llegar al módulo Bluetooth con las premisas ya expuestas es de:

$$T_{Sensorización+Envio} = \sum (T_{Sensorización} + T_{envio}) = 435\mu s + 1.51ms = 1.95ms$$

Es decir, que podríamos mejorar bastante el ancho de banda del sistema.

5.3. Wi-Fi comprobación.

En este caso se ha realizado la misma comprobación que para Bluetooth pero para 10 segundos, obteniendo:

```
+IPD,0,441:GET / HTTP/1.1

Tiempo comienzo lectura petición: 34125 ms
Tiempo final lectura petición: 34250 ms
Tiempo final de envio: 34625 ms


0,CONNECT



+IPD,0,467:GET / HTTP/1.1

Tiempo comienzo lectura petición: 44625 ms
Tiempo final lectura petición: 44750 ms
Tiempo final de envio: 45000 ms


0,CONNECT



+IPD,0,467:GET / HTTP/1.1

Tiempo comienzo lectura petición: 55125 ms
Tiempo final lectura petición: 55125 ms
Tiempo final de envio: 55375 ms
```

Ilustración 38: Comprobación resultado Wi-Fi.

Como se puede observar el sistema no avanza cíclicamente cada 10 segundos, esto se debe a que el Navegador Web empezará a contabilizar los 10 segundos una vez obtenida y actualizada la Web. Es decir, el transcurso entre enviar la petición, tratarla por el Servidor y la recepción con el propio tratamiento por el Navegador Web no se ha tenido en cuenta y repercute en aproximadamente 375ms de desfase, que se podrían ajustar para acercarlo a lo ideal.

No se ha hecho el apartado de resultados porque no llega a ser muy eficiente.

6. Mejoras y comentarios.

- Mejorar el Timer en el Cliente Wi-Fi, obteniendo el tiempo global del Navegador Web y actuar en consecuencia para sincronizar el sistema correctamente.
- Implementar mediciones a la salida de la Uart, mediante ModelSim, para obtener una medición experimental del ancho de banda y no teórica.

- Integrar un sistema operativo dentro del microprocesador y dentro de éste un servidor ya diseñado.
- Analizar donde está el cuello de botella en el envío de datos mediante bluetooth y wifi e intentar mejorarlo mediante mejoras Hardware o Software, o una combinación de éstas.
- Mejorar la interfaz gráfica de la web y de la app.

7. Presupuesto.

Elementos:

Tabla 21: Presupuesto elementos.

Ref	Ud	Descripción	Precio	Cantidad	Total
P0082	ud	DE0-Nano] DE0-Nano Development and Education Board (Academic)	61 $	1	61 $
P0499	ud	RFS] The RFS Card	59 €	1	59 €

Total:

Tabla 22: Presupuesto final.

Cantidad total elementos:	$120
Coste envio:	$38.49
Coste final:	**$158.49**

8. Archivos.

Podemos distinguir dos tipos de archivos:

Los suministrados por Terasic, en el cual podemos encontrar ejemplos tanto de la placa DE0-Nano como de la RFS, además de manuales de usuario de estas, como hojas de características de los elementos que lo componen.

Se encuentra en:

- RFS_v.1.2.0_SystemCD.
- DE0-Nano_v.1.2.2_SystemCD

Por otra parte podemos encontrar los archivos específicos de este trabajo:

- Bluethooth1_0, se encuentra el diseño hardware y software del servidor de Bluetooth.
- Android_Project, se encuentra el proyecto de la aplicación Android de Eclipse y la aplicación l compilada llamada RFS.
- Proyecto1_0, se encuentra el diseño hardware y software del servidor y del cliente Wi-Fi.

9. Videos.

Bluetooth:

- https://www.youtube.com/watch?v=vaMVgQrqGAo

Wi-FI:

- https://www.youtube.com/edit?o=U&video_id=hpxFT5mcMGo

10. Conclusiones.

Como conclusión, voy a empezar con la experiencia personal que me ha reportado realizar este Trabajo. Considero que es un trabajo transversal donde posiblemente no se profundiza en ninguna materia pero que me ha permitido obtener una visión general y bastante bien enfocada de cómo funciona tecnologías como la Bluetooth o Wi-Fi y como se relacionan con los servidores y clientes. Además de estudiar nuevas tecnologías para mí como el protocolo Http o lenguajes de programación como Android(Java), JavaScript, lenguajes de marcas como Html o XML. Todo este compendio de tecnologías más la referentes a desarrollo en fpga, se han aunado en un mismo proyecto lo que me ha repercutido, como ya he mencionado, en obtener un visión general muy certera.

En cuanto a los resultados del trabajo, puedo decir que estoy satisfecho con la comunicación mediante Bluetooth por la velocidad y la sincronía, no puedo decir lo mismo en cuanto al Wi-Fi, aunque considero bastante factible mejorarlo.

11. Referencias.

- Apuntes asignatura Sistemas Integrados – Master Universitario en Ingeniería de Sistemas Electrónicos.
- Apuntes asignatura Sistemas Embebidos – Master Universitario en Ingeniería de Sistemas Electrónicos.
- Apuntes asignatura Comunicaciones Industriales – Master Universitario en Ingeniería de Sistemas Electrónicos.
- Android, Guía de desarrolladores, Anaya – Frank Abelson, Charlie Collins, Robi Sen.
- HTML5, Anaya –Alonso Álvarez Garcia.
- JavaScript, Anaya –Terry MvNage.